Wireless Pers Commun (2016) 91:1681–1695
DOI 10.1007/s11277-016-3355-0

An Efficient Device Authentication Protocol Without Certification Authority for Internet of Things

Sunggyun Jang[1] · Ducsun Lim[1] · Jinyeong Kang[1] · Inwhee Joe[1]

Published online: 11 May 2016

Abstract Wireless network devices are used for the Internet of Things in a variety of applications, and although the IoT has many benefits, there are some security issues in this area. Hacking tools that are widely used in wireless communication enable the attacker to export the information stored in the device memory. Devices within the IoT should not allow this information to be accessed without an authentication. In this paper, we propose an efficient device authentication protocol without certification authority for the Internet of Things. Compared to the existing Constrained Application Protocol, the proposed protocol increases efficiency by minimizing the number of message exchanges. Since our protocol is based on a keyed hash algorithm, the Certificate of Authority is not required. Experimental results show that the proposed authentication protocol improves the security level and reduces the resource consumption of devices.

Keywords Merkle Tree · Root Hash · MAC (Message Authentication Code) · IoT (Internet of Things) security · Authentication

1 Introduction

The device density per network is increasing rapidly. Many wireless protocols are used to create an IoT environment via protocols such as RFID, GPS, IEEE 802.11, Bluetooth, and zigbee. This is also accomplished through a variety of IT technologies. The number of devices used in 2009 is estimated to be approximately 900 million IoT, while in 2020 this number is projected at approximately 250 million devices [1–3]. As IoT technology grows, so does the possibility of information leakage, as most IoT devices are exposed to information leak and hacking. Numerous suggestions have been proposed in this regard, such as

✉ Inwhee Joe
iwjoe@hanyang.ac.kr

1 Department of Electronics and Computer Engineering, Hanyang University, Seoul, Korea

a security protocol in the IoT environment, Datagram Transport Layer Security (DTLS); or embedded Security Socket Layer (SSL), WolfSSL. DTLS is a protocol that provides security through datagram protocol communication, and requires six message packet exchanges. If a packet is lost, message packets must be transmitted from the beginning again, which may result in poor device performance in the limited IoT environment [4, 5]. WolfSSL is an embedded protocol based on SSL/TLS. This lightweight library can be used with resources of limited size and mobility; however, it is not easily applied in the case of small devices with few resources because it is not lightweight enough [6].

IETF has been categorized by a device resource that configures the IoT environment and is shown in Table 1. Currently, many IoT products have applied the IoT platforms. These platforms can benefit both the developer and the company, and offer a platform that brings many benefits, including:

- reduced development cycle of a project
- reduced development mistakes
- reduced cost

The platforms enable the developer to make new IoT devices easily and quickly. Platforms can make various devices in the areas of consumer and smart home, smart infrastructure, security and surveillance, healthcare, retail, industry, and transportation. The IoT device, Network, Security, and Service constitute the IoT platform. In the IoT platform, the IoT device collects various data and consists of a processor, wireless media, memory, and special sensors that sense row data. The processors used in the IoT devices that are included in Embedded Systems range from 8 to 32 bit. The M2M device generally uses the 8 bit microprocessor; however, the IoT device uses not only an 8 bit microprocessor but also a 32 bit microprocessor. The microprocessor makes it possible for an IoT device to connect to a variety of smart devices such as Smartphones and Tablets using wireless communication.

To support communication, the IoT device utilizes wireless media such as Wi-Fi and BLE. It is important for the IoT device to have a protocol stack software for each wireless media. The software is held within the IoT device memory using either ROM or Flash. Wireless media came into widespread use in IoT devices for its high-performance and ample memory. The IoT software is divided into five parts within memory, as shown in Fig. 1. Each SW must be included in the memory except the Security SW, as shown in Table 2.

The IoT device requires sufficient memory to gather, send, and receive sensor data from a smart device. IoT device platforms have been designed by four companies, ARM mbed, ATMEL Arduino, Raspberry Pi, and Intel Edision, which all offer the required 32 bit high microprocessor. ARM Mbed specifically has a 100 MHz and 32 bit Cortex-M processor. NXP, Freescale, TI, STMicro, and Nordic are based on Mbed, and the Arduino offers microprocessors ranging from 8 to 32 bit. The Raspberry Pi has an ARM1176JZF 700 MHz, 32 bit microprocessor. The Intel Edison has a 400 MHz and 32 bit Quark processor, as shown in Table 3.

Table 1 Specification for each class

	RAM (KB)	Flash (KB)
Class 0	$\ll 10$	$\ll 100$
Class 1	~ 10	~ 100
Class 2	~ 10	~ 250

Fig. 1 IoT platform

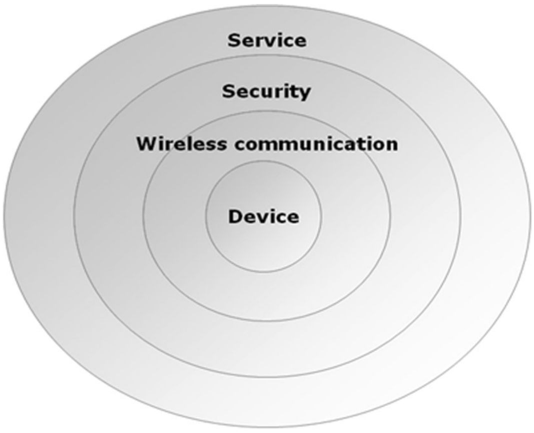

Table 2 IoT software category

Category	Required
Based SW	Mandatory
Protocol stack SW	Mandatory
Control the sensor and sensing	Mandatory
Security SW	Optional
Application	Mandatory

Table 3 IoT device speed for each processor

	CPU (MHz)
Mbed	~ 100
Arduino	$\ll 80$
Raspberry Pi	~ 700
Edison	~ 400

Also, ARM, ATMEL, Raspberry Pi, and INTEL have large memory such as RAM and Flash, as shown in Table 4.

Mbed and Arduino are used in various applications, such as in wearable devices with low performance applications, while Raspberry Pi and Edison can be used in high speed devices such as gateway with high applications. As mentioned above, the classes suggested by IETF are simple and can be applied in a limited environment. This study targets devices

Table 4 IoT device memory size for each platform

	RAM	Flash
Mbed	$\sim$256 KB	$\sim$2 MB
Arduino	$\sim$64 KB	$\sim$512 KB
Raspberry Pi	$\sim$256 MB	$\sim$1 GB
Edison	$\sim$1 GB	Dependent

Table 5 Specification for each weight

	RAM	Flash
Lightweight	≪64 KB	≪512 KB
Middleweight	~256 KB	~2 MB
Heavyweight	~1 GB	~16 GB

that have core roles; Table 5 presents three classes categorized by the size of their resources.

This study is focused on the middleweight among the three classes and suggests a sporadic authentication protocol using the Hash Tree if authentication among devices is difficult, due to the absence of a central control server. The flash memory size required for the suggested protocol is approximately 5000 KB and the testing device has stm32 cortex m4, which is a middleweight resource.

This paper is configured as follows: Sect. 2 demonstrates the relevant studies, Sect. 3 demonstrates working principles of the suggested protocol, and the conclusion is presented in Sect. 4.

2 Related Works

2.1 Merkle Tree

The Hash Tree first suggested by Merkle is configured, as shown in Fig. 2. Each attribute was hashed to form a binary tree and the message is verified using a Root Hash.

$$P[i,j] = f(P[i,(i+j-1)/2] \parallel P[(i+j+1)/2,j]) \tag{1}$$

Equation (1) calculates P(1) and shows the process of assigning a space for mapping the nodes with an attribute value or hash value, respectively. Equation (2) shows a computation that requires two child trees to derive the Root Hash.

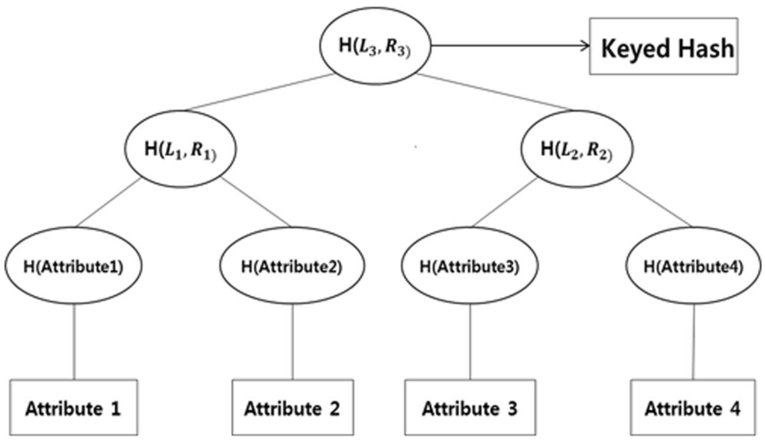

Fig. 2 Merkle Hash Tree

$$P^{n_{parent}} = f(P^{n_{left}} \parallel P^{n_{right}}) \tag{2}$$

This is called a Tree Authentication [7]. The most important attribute at this point is the authentication of the Root Hash and the Merkle Tree proof that the correct Root Hash value is with the authenticator. To apply this in an IoT environment, the central control server must have all the Root Hash values of every user that causes issues with memory space and verification. Even when configured this way, the system is still exposed to hacking and information leaks [8, 9].

2.2 Specification of CoAP

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained (low-power, lossy) networks. The nodes often have 8-bit microcontrollers with small amounts of ROM and RAM, while constrained networks such as IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs) often have high packet error rates and a typical throughput of 10 s of KB/s. The protocol is designed for machine-to-machine (M2M) applications such as smart energy, and building automation CoAP provides a request/response interaction model between application endpoints, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and Internet media types. CoAP is designed to easily interface with HTTP or for integration with the Web while meeting specialized requirements such as multicast support, very low overhead, and simplicity for constrained environments.

In this case, the sender and receiver share a secret key K, which they will use to authenticate their transmissions. We describe the message authentication goal and various methods of achieving it. Issues which are still being resolved will be explicitly noted in this text, as shown in Table 6.

Let us examine some example message authentication codes and use the definition to assess their strengths and weaknesses. We fix a PRF, as shown in Table 7.

Table 6 Definition authenticity of an encryption scheme

Let = (K, E,D) be an encryption scheme and let A be an adversary.
We consider the following experiment:
Experiment: Exp auth(A)
K = K $
Run K is A_{EK} $\odot$, V_{FK} $\odot$ where V_{FK}(C) is 1 if D_K(C) $\in \{0,1\} *$
and 0 if D_K(C) = $\perp$ (return value)
if A made a V_{FK} query C such that
? The oracle returned 1, and
? A did not, prior to making verification query C,
make an encryption query that returned C,
then return 1 else return 0,
The authenticity advantage of A is defined as
$Adv(\prod auth(A)) = Pr[Exp(\prod auth(A)) \rightarrow 1]$

Table 7 Definition authenticity of an encryption scheme

F: K {0, 1} n → {0, 1} n.
Our first scheme
MAC1: K {0, 1}∗ → {0, 1}∗ algorithm MAC(M)
algorithm $MAC1_K(M)$
if (|M| mod n 6 = 0 or |M| = 0) then return ⊥
Break M into n-bit blocks M = $M_1 \ldots M_m$
for i ← 1 to m do $Y_i F_k(M_i)$
T ← $Y_1 \oplus \cdots \oplus Y_n$
return T

2.3 C.Message Authentication Code (MAC)

For many people, privacy is the goal most strongly associated with cryptography; but message authentication is arguably even more important. Indeed you may or may not care if some particular message you send remains private, but you almost certainly want to be sure of the originator of each message that you act on. Message authentication is what buys you that guarantee. Message authentication allows one party, the sender, to send a message to another party, the receiver, in such a way that the receiver will almost certainly know if the message is modified a route. Message authentication is also called data-origin authentication, and is said to protect the integrity of a message, ensuring that each message that is received and deemed acceptable is arriving in the same condition that it was sent out with no bits inserted, missing, or modified. Here we will be looking at the shared-key setting for message authentication (remember that message authentication in the public-key setting is the problem addressed by digital signatures). The AES and SHA algorithm is widely used in MAC or an authentication [6, 8, 9].

Authentication protocol in current IoT environments is shown in Fig. 3.

(1) Initial U (user or sensor node) tries to request authentication with the Cipher Suite options, at which time the Cipher Suite as a set of encryptions of RSA, SHA, or AES occasionally selects and uses an arbitrary encryption scheme.
(2) V (server or sensor node) notifies that it is ready to authenticate when a request is received.
(3) U with a reply from V forms a message tag with a private key shared with random seeds presented with N. U forms a MAC and transmits to V using the authentication, private key, and message tag.
(4) V verifies the validity of the MAC and tests the validity of the authentication and private key shared by U and V, then forms a MAC in the same process and transmits.
(5) U verifies the validity of the MAC received from V and the authentication is finalized if there is no problem [10].

As shown in the abovementioned processes, authentication through MAC is simple and has relatively fewer hand-shakes; however, it requires devices with heavyweight resources in forming MACs and validation verification [11].

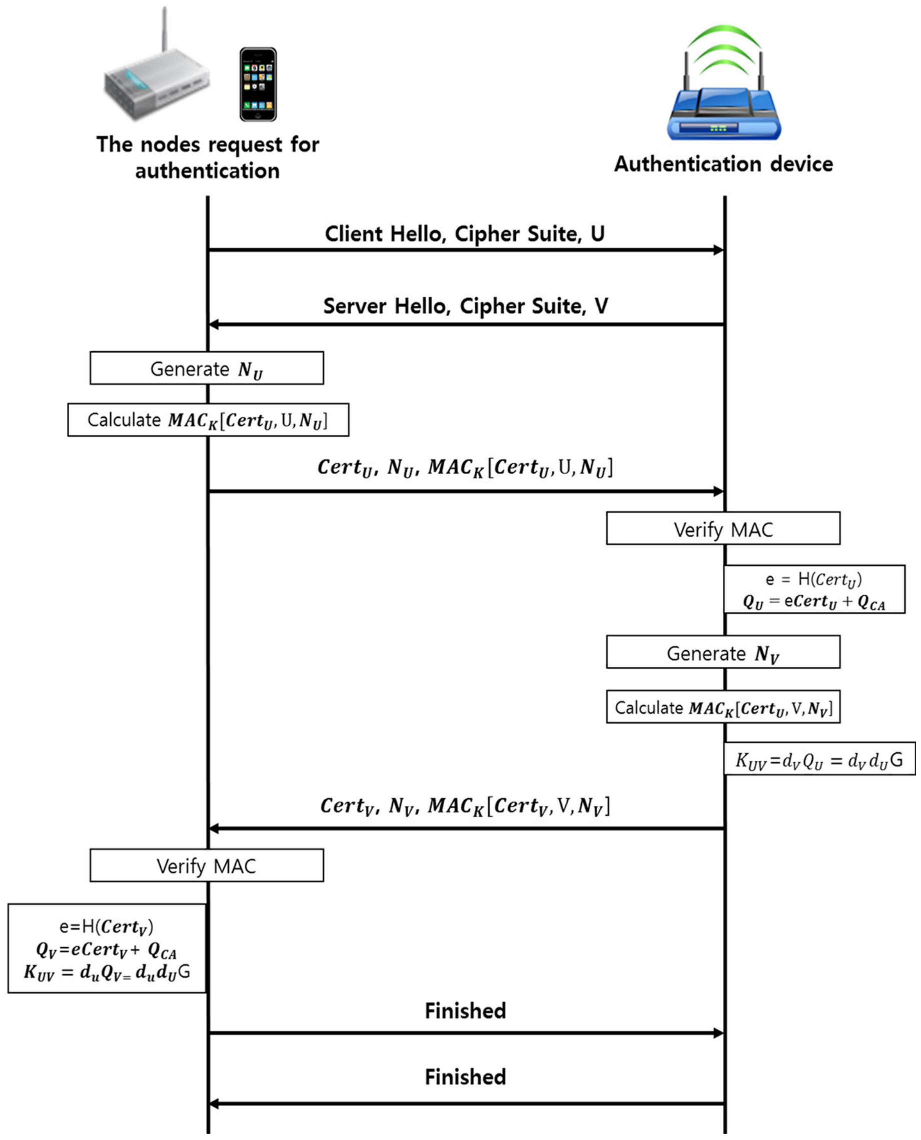
The nodes request for authentication
Authentication device
Client Hello, Cipher Suite, U
Server Hello, Cipher Suite, V
Generate N_U
Calculate $MAC_K[Cert_U, U, N_U]$
$Cert_U$, N_U, $MAC_K[Cert_U, U, N_U]$
Verify MAC
e = H($Cert_U$)
$Q_U = eCert_U + Q_{CA}$
Generate N_V
Calculate $MAC_K[Cert_U, V, N_V]$
$K_{UV} = d_V Q_U = d_V d_U G$
$Cert_V$, N_V, $MAC_K[Cert_V, V, N_V]$
Verify MAC
e=H($Cert_V$)
$Q_V = eCert_V + Q_{CA}$
$K_{UV} = d_u Q_V = d_u d_U G$
Finished
Finished

Fig. 3 Flow diagram of authentication phase

3 Proposed Authentication Protocol

As discussed in the previous section, classes suggested by IETF are lightweight in this paper, as shown in Table 5. The lightweight class is small enough to include the IoT software in memory as shown in Table 2. The middleweight class in this paper is suitable for devices in applications, such as wearables for SmartHome and Healthcare applications, because the middleweight class has enough memory to embed in ash memory. The middleweight class makes it possible for the device to contain the IoT software in memory, as shown in Table 4. Although middleweight devices are low in price, the class exhibits high-performance and meets the requirements suggested by IETF. Here is a purposed Keyed Hash algorithm, which is required from lightweight to middleweight classes because the IoT device should support the 32 bit microprocessor and have a memory size of at least 512 KB. The suggested protocol uses a sporadic authentication at the time that an authentication is required in order to address specific issues. It is able to authenticate between devices with limited recourses in the absence of central control.

(1) The Device (device that requires the authentication) transmits either a MAC address or a Serial number to the Target Device (device that performs the authentication). A MAC address and Serial number is encryption using AES as show in Fig. 4 in first step.

(2) The Target Device uses the leftmost node value as a MAC or serial and reflects a Hash value to the parent node through four encryption rounds (a set of encryptions of RSA, SHA, or AES occasionally selects). The right node is a Time Stamp that is used to form a Hash Tree. After the Hash Tree is formed, the Target Device sends the Time Stamp on the rightmost node to the device via four encryption rounds. It

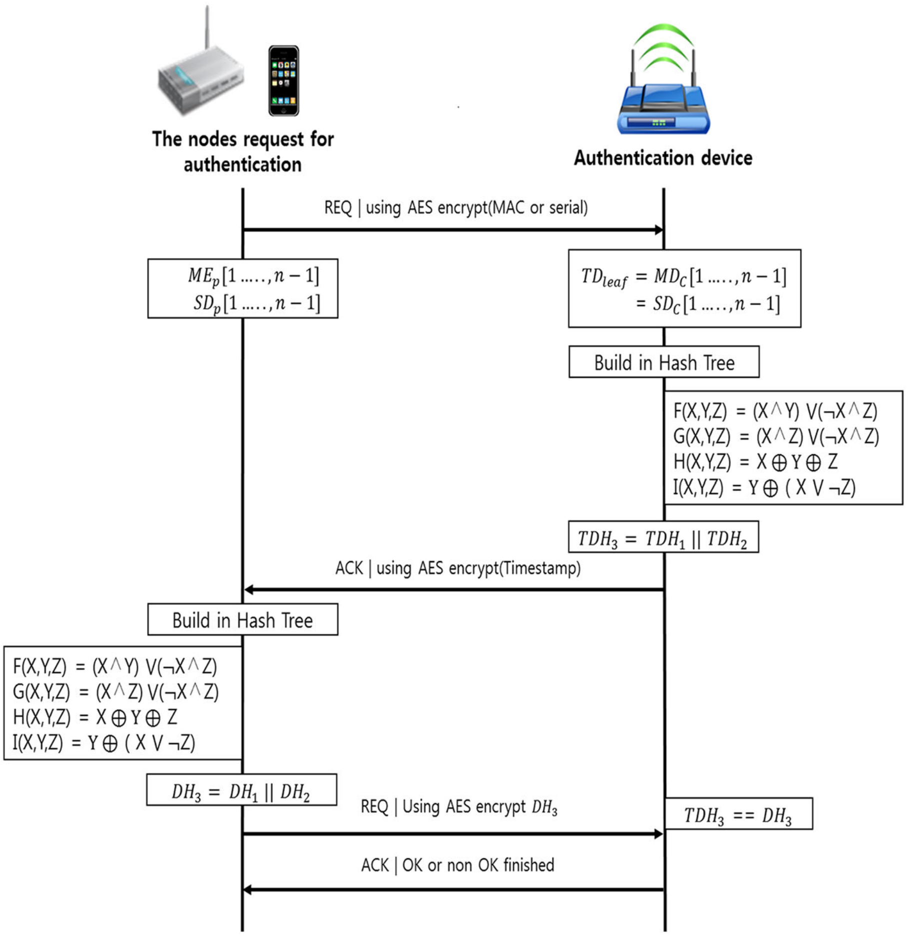
The nodes request for authentication
Authentication device
REQ | using AES encrypt(MAC or serial)
$ME_p[1 \ldots\ldots, n-1]$
$SD_p[1 \ldots\ldots, n-1]$
$TD_{leaf} = MD_C[1 \ldots\ldots, n-1] = SD_C[1 \ldots\ldots, n-1]$
Build in Hash Tree
F(X,Y,Z) = (X ∧ Y) V(¬X ∧ Z)
G(X,Y,Z) = (X ∧ Z) V(¬X ∧ Z)
H(X,Y,Z) = X ⊕ Y ⊕ Z
I(X,Y,Z) = Y ⊕ (X V ¬Z)
$TDH_3 = TDH_1 \mid\mid TDH_2$
ACK | using AES encrypt(Timestamp)
Build in Hash Tree
F(X,Y,Z) = (X ∧ Y) V(¬X ∧ Z)
G(X,Y,Z) = (X ∧ Z) V(¬X ∧ Z)
H(X,Y,Z) = X ⊕ Y ⊕ Z
I(X,Y,Z) = Y ⊕ (X V ¬Z)
$DH_3 = DH_1 \mid\mid DH_2$
REQ | Using AES encrypt DH_3
$TDH_3 == DH_3$
ACK | OK or non OK finished

Fig. 4 Flow diagram of proposed protocol

can be seen below that less calculations are required than for those shown in Fig. 3. The flow is as follows:

$$\begin{aligned} F(X,Y,Z) &= (X \wedge Y) \vee (\neg X \wedge Z) \\ G(X,Y,Z) &= (X \wedge Z) \vee (\neg X \wedge Z) \\ H(X,Y,Z) &= X \oplus Y \oplus Z \\ I(X,Y,Z) &= Y \oplus (X \wedge \neg Z) \end{aligned}$$

F and G are functions (X, a logical product of Y) and (logical AND of NOT X and Z) to logical sum operation, the H function shows the XOR operation for all arguments, and the I function (X-OR and NOT Z) indicates the XOR operation in the Y operation.

(3) The Device from the Hash Tree is formed at the Device in the same way that the Hash Tree is formed at the Target Device. The device and target device has the same Hash Tree algorithm and four encryption rounds. Then two device has each TDH3 and DH3.

(4) The Root Hash is transmitted for authentication of its validity at the Device, the Target Device authenticates, and the Hash shake is completed. (Device: D, Target Device: TD, DH3: Root Hash)

In Table 8, the message transmission was encrypted by AES128 bit, (3) encrypts the key information via DH3 AES. This has the advantage of double encryption (DH1-3: Device Hash 1–3, TDH1-3: Target Device Hash 1–3 as shown in Fig. 2).

When the Root Hash for authentication is formed, DH1 and DH2, the left and right Hash values, respectively, take up only 64 bits to form the Root Hash, enabling high security. This security results from one of the tMac Address, Serial number, or Time Stamp being snipped while the message is being transmitted. Figure 5 is a graphic of the formula shown in Tables 8 and 9 and shows the differences apparent to the conventional Merkle Hash Tree, This is used for authentication of attribute 1–2, consisting of a MAC address and Serial number unique identification tag device, so that each forms a 128 bit empty bit padding. Values used in padding are derived via a function of the time seed and argument. In this way, the padding has 128 bits of Message generating a Hash value with the SHA128 algorithm, L1H1 is a Hash generated using the MAC address (LeftHash), and a Hash produced using the Serial number is R1H2 (RightHash). The Hash values of both the generated 128 bit top of the L1H1 64 bit, and R1H2 in the sub 64 bit Nomitori, the padding without 128 bit after setting, is Hashing to again SHA128 algorithm. This is used as the generated Hash value to authenticate as Keyed Hash.

We use the definition of the tagging algorithm to see that

Table 8 Message transport sequence

(Device: D, Target Device: TD, DH3: Root Hash)
(1) D → TD: Mac address or serial number
(2) TD → D: Time Stamp
(3) D → TD: DH3

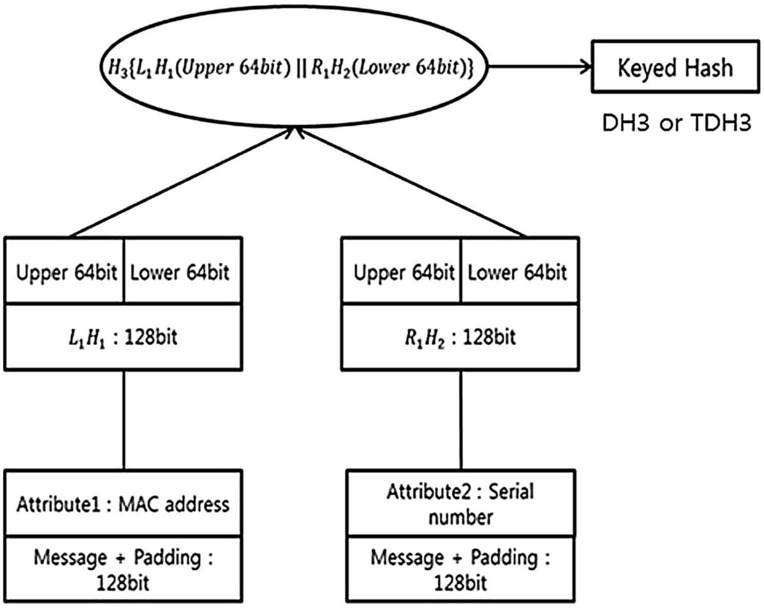
$H_3\{L_1H_1(Upper\ 64bit) \| R_1H_2(Lower\ 64bit)\}$
Keyed Hash
DH3 or TDH3
Upper 64bit
Lower 64bit
L_1H_1 : 128bit
Upper 64bit
Lower 64bit
R_1H_2 : 128bit
Attribute1 : MAC address
Message + Padding : 128bit
Attribute2 : Serial number
Message + Padding : 128bit

Fig. 5 Root Hash procedure flow

Table 9 Create a Root Hash procedure

(1) DH3 (Root Hash) = DH1 ‖ DH2
(2) TDH3 (Root Hash) = TDH1 ‖ TDH2

$$T1 = F_k(SD_p \text{and} \mathrm{ME_p}) \oplus \mathrm{F_K(k\_SHA2)}$$
$$T2 = G_k(TD_{leaf}) \oplus F_K(SD_p \text{and} \mathrm{ME_p}) \oplus \mathrm{F_K(k\_SHA2)}$$
$$T3 = G_K(DH1 \parallel DH2) \oplus F_K(k\_SHA2)$$

T1, T2, and T3 are computed for each sequence in Fig. 4. T1, SDp, and Mep are used in the argument of the F function, and the last argument is Fig. 4. Padding values as described in X are used. The Hash values derived as a function F are calculated again to transmit the sequence SHA128. T2 is the leaf node of the Target Device as an argument of G functions, that is, using the time stamp and SDp and Mep, and then transmits the double-encrypted message with SHA128, as with T1. T3, the target device to authenticate the device seeking transmission, indicates whether the sequences of each Keyed Hash match and SHA128 are set to double encrypt by all processes in order to transmit a sequence of strengthened security. The SHA128 encrypted sequence is set to the value obtained by decoding, even if the sniffing is a Hash value, and the security is a difficult surface to analogize so that the original data is strong.

4 Performance Evaluation

There are three performance evaluations. First, the authentication delay time of each platform is compared with the initial authentication and re-authentication. Second, the code size of each platform is compared with the code size before compiling the codes. Last, CoAP and the keyed hash are compared with the amount of power consumption required using the Wi-Fi module in to authenticate through a significant number of cycles.

4.1 Authentication Delay Time of Each Platform

Performance was compared to the code size of each of the platforms by measuring the speed and the effectiveness of the Keyed Hash, as confirmed through verification. Figure 4 shows the time of the initial authentication and the time required to perform the re-authentication. WolfSSL, CoAP, and MQTT exhibited a performance speed between 150,000 and 230,000 ms in Fig. 6. Further, the value stored in the certificate for re-authentication when performing according to all three protocols was reduced to a width of 50,000 m, whereas the proposed algorithm is required because it uses the Keyed Hash one-off and disposes of the authentication and re-authentication at the same time.

4.2 Code Size of Each Platform

Table 10 shows the code size to be used only for authentication on each protocol. This has decreased to approximately 1/5 to 1/3 the size of the existing protocols code, making it lightweight.

4.3 Power Consumption

For calculating the power consumption of the keyed hash, we used Wireless 802.11n with the wireless LAN module for the device and the target device as shown in Table 10. We compute the TX and RX power consumption for only send and receive times, except when executing the keyed hash algorithm.

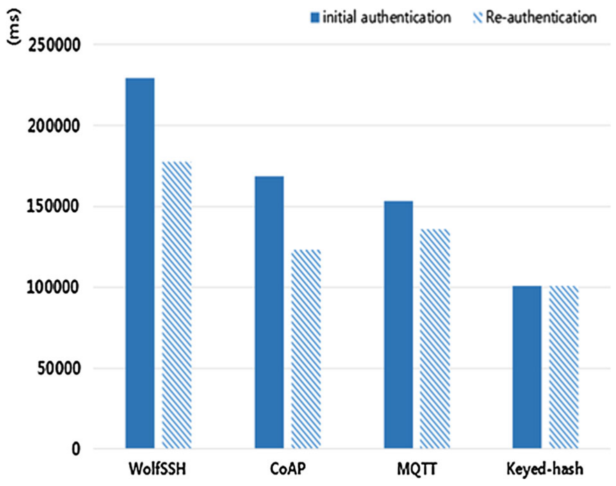

Fig. 6 Authentication delay time of each platform

Table 10 Code size of each platform

Platform	Code size	Units
WolfSSL	31,259	KB
CoAP	27,118	KB
MQTT	16,875	KB
Keyed Hash	4900	KB

Equations (3) and (4) are that Txp and Rxp are typical values and Message/BurstTx is transmission time, N is number of messages.

TX power consumption can be given as,

$$P_{TotalTx} = Txp \cdot \frac{Message}{BurstTx} \cdot N \tag{3}$$

Let us assume that there is Tp TX power and Message authentication data and N number of messages.

The RX power consumption can be given as

$$P_{TotalRx} = Rxp \cdot \frac{Message}{ContinuousRx} \cdot N \tag{4}$$

A key hash is needed with two times TX and RX, as shown in Fig. 3; however, CoAP has three times TX and RX, as shown in Table 11. Here, we compare the Keyed Hash and CoAP.

4.4 Quantative Analysis

The proposed keyed hash scheme protocol, even without the presence of a Certificate Authority (CA), has the conspicuous advantage of being able to perform authentication. This is a complex procedure where the performance and power consumption are more efficient than the existing system; this occurs when you place a large specific gravity on the role of the CA. This system also has the advantage of solving some of the security problems, as shown in Fig. 7.

For example, if the server with the CA role is set to save all of the information about the certificate and the server is hacked, you might have a significant leak of information. If the protocol that provides this and subsequent authentication uses the quick discard method, security is enhanced. Moreover, when using an encryption algorithm in duplicate for every sequence operation, it is difficult to analogize the original data; by modifying the conventional keyed hash method for use, it is possible to provide enhanced security against hackers.

Table 11 RX and TX power consumption using Wi-Fi

Power	Description	Typical	Units
TX consumption	Burst TX (150 Mbps)	439	mW
RX consumption	Continuous RX (150 Mbps)	552	mW

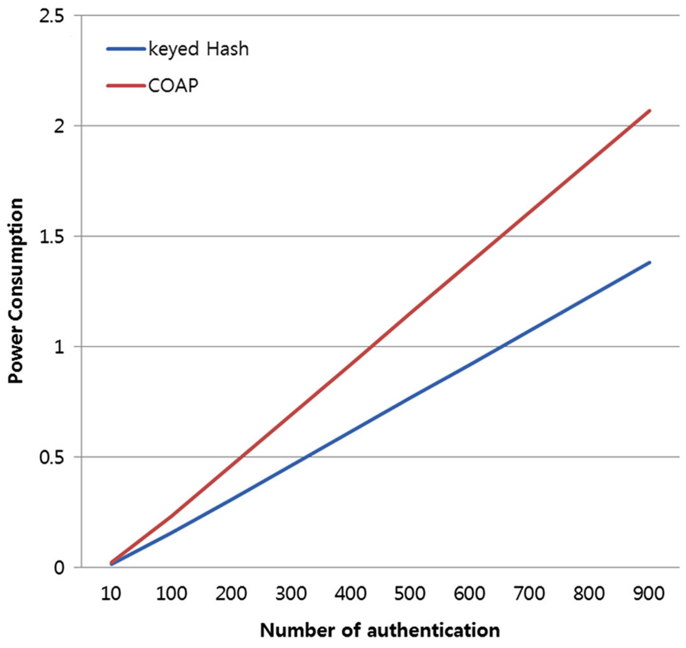

Fig. 7 Compare with power consumption

5 Conclusion

This paper suggests a new authentication protocol that enables authentication between devices in the absence of a central control server based on a keyed hash. The commonly used Hash-Tree authentication has been analyzed from various angles and applied, lending the advantages of being able to authenticate with less resources, fewer hand-shakes, and reduced information leaks in sporadic authentications. The proposed keyed hash scheme protocol without the need for the presence of a Certificate Authority (CA), has the advantage of being able to stand out an authentication. This is a complex process in which the performance and power consumption are more efficient than the conventional method in terms of the number of parts, and solves the security problems that occur when a significant amount of the weight is assigned to the role of the left CA. If the server that serves the CA and stores all of the information about the certificate is hacked, information leakage may occur, as shown in Table 12.

In this protocol, the proposed method of using the waste immediately after authentication was added as additional security. In addition, it is difficult to infer the original

Table 12 Comparison of CoAP and proposed protocol

	CoAP	Proposed
CA	Need	None
Security level	Single encryption	Double encryption
Power consumption	2.0691	1.3794
Code size	27,118	4900

because of the use of a Data Encryption Algorithm in every action sequence with a double, and because the existing keyed hash method is more secure from invaders. Further study is needed so that this method can be applied in diverse environments such as BLE or zigbee, which are continuously evolving. Another area of focus for future research will be the Key generator and Key distribution for the security platform.

References

1. Khan, B. H. (2000). *A framework for web-based learning*. Englewood Cliffs, NJ: Educational Technology Publications.
2. Zhou, H. (2010). *Web 4.0: The "Chinese Style" definition of Internet of Things [Z/OL]*. http://www.wlw.gov.cn/zxzx/wldt/594260.
3. Jacobs, I. S., & Bean, C. P. (1963). Fine particles, thin films and exchange anisotropy. In G. T. Rado & H. Suhl (Eds.), *Magnetism* (Vol. III, pp. 271–350). New York: Academic.
4. Park, J., & Kang, N. (2014). Lightweight secure communication for CoAP-enabled internet of things using delegated DTLS handshake. In *International conference on information and communication technology convergence (ICTC)* (pp. 28–33).
5. Nicole, R. (1987). Title of paper with only first word capitalized. *J. Name Stand. Abbrev,* 740–741.
6. Guo, J., Peyrin, T., Poschmann, A., & Robshaw, M. (2011). The LED block cipher. In *Cryptographic hardware and embedded systems CHES 2011, LNCS* (Vol. 6917/2011, pp. 326–341). Springer.
7. Merkle, R. C. (1989). A certified digital signature. In *CRYPTO, volume 435 of lecture notes in computer science* (pp. 218–238).
8. Lipmaa, H. (2002). On optimal hash tree traversal for interval timestamping. In *Proceedings of information security conference*.
9. Klintsevich, E., Okeya, K., Vuillaume, C., Buchmann, J., & Dahmen, E. (2007). Merkle signatures with virtually unlimited signature capacity. In *5th international conference on applied cryptography and network security—ACNS07*.
10. Brachmann, M., Keoh, S. L., Morchon, O., & Kumar, S. (2012). End-to-end transport security in the ip-based internet of things. In *Proceedings of the 21st international conference on computer communications and networks (ICCCN)* (pp. 1–5).
11. Raza, S., Trabalza, D., & Voigt, T. (2012). 6LoWPAN compressed DTLS for CoAP. In *Proceedings of 8th IEEE conference on distributed computing in sensor systems (DCOSS)* (pp. 287–289).

Sunggyun Jang received his the B.S. degrees in Department of Computer Engineering from Tongmyong University, Busan, Korea in 2003 and he received M.S. degrees in Department of Computer Engineering from Hanyang University, Seoul in 2009. Now, he has been studying for Ph.D. degrees in Hanyang University, Seoul, Korea. His research area includes Embedded Software, IoT device security.

Ducsun Lim received his the B.S. degree from Department of Computer Engineering, Hanyang Cyber University, Seoul, Korea in 1998. Now, he has been studying for M.S. degree in Wireless Mobile Network Laboratory, in Computer and Software from Hanyang University, Seoul, Korea. His current research interest include SDN, IoT and cloud computing.

Jinyeong Kang received his the B.S. degrees from the Department of Computer Engineering, Hanyang University, Seoul, Korea in 1998. Now he has been studying for M.S. degree in Wireless Mobile Network Laboratory, Hanyang University. His research area includes openstack, SDN, clouding computing and IoT.

Inwhee Joe received his B.S. and M.S. degrees in Electronics Engineering from Hanyang University, Seoul, Korea, and his Ph.D. degree in Electrical and Computer Engineering from Georgia Institute of Technology, Atlanta, GA in 1998. Since 2002, he has been a faculty member in the Division of Computer Science and Engineering at Hanyang University, Seoul, Korea. His current research Interests include mobile internet, cellular system and PCS, wireless sensor networks, mobile ad-hoc networks, multimedia networking, performance evaluation.