ESSE: Efficient Secure Session Establishment for Internet-Integrated Wireless Sensor Networks

Namhi Kang, Jiye Park, Hyeokjin Kwon, and Souhwan Jung

1Duksung Women’s University, Seoul 132-714, Republic of Korea
2Soongsil University, Seoul 156-743, Republic of Korea

Correspondence should be addressed to Souhwan Jung; souhwanj@ssu.ac.kr

Received 9 January 2015; Accepted 12 March 2015

Copyright © 2015 Namhi Kang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The dramatically increasing number of connected things based on Internet protocol is leading to a new concept called the Internet of Things (IoT). The Internet-integrated wireless sensor network has recently become one of the most important service targets in IoT field. To provide secure IoT services, the IETF proposed using Datagram Transport Layer Security (DTLS) as a de facto security protocol. In this paper, we examine problems that can happen when applying the DTLS protocol to the IoT, which comprises constrained devices and constrained networks. To solve the problems at hand, we separate the DTLS protocol into a handshake phase (i.e., establishment phase) and an encryption phase (i.e., transmission phase). Our approach enhances the performance of both device and network by using a way to delegate the DTLS handshake phase. The proposed scheme supports secure end-to-end communication despite using delegation.

1. Introduction

Continuous innovation in information and communications technologies has led to a new concept of everything being internetworked, which is called the Internet of Things (IoT). Industrial and research organizations predict that tens of billions of everyday objects will be interconnected with each other at any time and place to support humans in the near future [1, 2]. Such interconnected things automatically discover a target service for humans and intelligently support the service by cooperating with nearby things without human intervention. Consequently, a smart and intelligent world can be realized.

The IoT has characteristics distinct from those of existing Internet environments. First, the IoT comprises resource-constrained devices, where the resources include CPU, memory, and a battery. The devices are classified into three categories according to their resource constraints [3]. In particular, devices in Class 0 are highly constrained (e.g., small motes). They cannot communicate with the Internet directly, so Class 0 devices have to get help from infradevices, such as a proxy or gateway. The constrained devices in the IoT can be a client or a server depending on the service scenario. In addition, the network is also constrained. To use battery power efficiently, small devices in the IoT adopt low-power wireless communications specifications like IEEE 802.15.4, which has a 127-byte maximum transmission unit (MTU) size. Because of the MTU size, a large-sized packet must be fragmented into small pieces at the link layer [4]. In this paper, we consider such a low-power and lossy network (LLN) as a connection between constrained embedded devices (i.e., for Internet-integrated wireless sensor networks).

Considering the above characteristics, the Constrained RESTful Environments (CoRE) Working Group (WG) of the IETF specified the Constrained Application Protocol (CoAP) [5]. CoAP is based on the User Datagram Protocol (UDP) and supports simple methods like GET, PUT, POST, and DELETE. It allows small devices to communicate in an efficient manner. Therefore, heterogeneous devices can be deployed into an IoT domain, such as smart cars, smart homes, and smart healthcare systems.

However, the increase in the number of devices connected to the Internet carries various potential risks as great as the possibility of making a smart world. Specifically, it increases
the number of targets that can be attacked. Because the IoT aims for on-demand service, it can have more private and personal information than any other services. One more important fact is that the capabilities of attacking systems are getting better and better as time goes by, unlike the limited resources of IoT devices. Also, such resource-constrained and heterogeneous devices are dynamically connected with each other according to the user’s position and service scenario as shown in Figure 1. As a result, both a mutual authentication method and a secure communication method (i.e., confidentiality and integrity) are highly required.

The CoRE WG proposed binding the DTLS protocol to CoAP as a secure communications protocol [6]. However, DTLS was not designed for the IoT but for traditional systems on the Internet. So there are several problems in applying it to the IoT directly. When using DTLS protocol, before sending encrypted data over a secure session, six flight handshake messages should be exchanged between entities. In low energy communication environments, message packets are fragmented into the 127-byte MTU size. These fragmented packets cause data loss and delay in an LLN, which leads to constrained network overhead [7]. Using a cipher suite based on certificates makes LLN conditions much worse. Furthermore, these handshake message flights can be a burden to devices in Classes 0 and 1 in the fact that such devices are vulnerable to denial of service (DoS) attacks [4].

To solve these problems, we propose and implement an “Efficient and Secure Session Establishment (ESSE)” system that separates the handshake phase and the data encryption phase under the DTLS protocol by using delegating modules. With the DTLS delegation process performing the handshake phase, several problems in an LLN (like data loss and delay) can be solved. Additionally, DoS attacks can be reduced because a constrained device performs the encryption phase only. Implementation results show that the proposed ESSE allows a constrained sensor to reduce energy consumption.

Our evaluation shows that the power consumption required to process DTLS handshake is more than nine times higher than the consumption of record layer processing. Also, ESSE requires small size of memory. The size of object implemented for a constrained node is about 13.8 Kbytes.

Part of the proposed system was published at an IEEE international conference in [8]. Two major extensions of this paper are the secure bootstrapping scheme for a resource-constrained node (Section 4) and implementation and evaluation of the proposed system (Section 6). The other parts of this paper are organized as follows. Section 2 provides an overview of related works. Section 3 describes the conceptual model of the proposed system. In Section 5, a means to establish a DTLS session is presented in detail. In Section 7 we discuss the security considerations of our proposed system, and then we conclude in Section 8.

2. Related Works

The DTLS protocol was designed for unreliable transmissions that use UDP and the DTLS consists of four protocols: Handshake, Alert, Change Cipher Spec, and Record. To avoid message fragmentation in the network layer, the DTLS protocol provides message fragmentation at the handshake layer. However, it is difficult to avoid such fragmentation in the network layer because the source does not know the MTU of the network of its receiver in advance. Also, a large amount of message must be sent during the DTLS handshake phase. For example, when the TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 cipher suite based on a raw public key certificate is used, six flight messages are fragmented into 27 datagram packets [12]. These fragmented packets induce problems, such as data loss rate increases and delays generated by packet retransmission; therefore, LLN conditions are aggravated. Besides, to reassemble a fragmented message packet, devices have to keep fragmented pieces of the message.
in the buffer until all the pieces arrive. This is a burden to a memory-constrained device. To solve these problems, several approaches have been proposed, but clear solutions have not yet been proposed. In this section, we review the proposed systems, especially to solve the problems that can occur when DTLS is applied to an IoT environment.

To reduce the burden on memory of the resource-constrained devices in the IoT environment, lightweight DTLS was proposed in [9]. Lightweight DTLS reduced the DTLS code size in order to lower the burden on constrained memory of the devices. Raza et al. proposed a method to reduce overhead in an LLN [10]. This proposed system reduces the transmitted message size by utilizing a technique to compress the DTLS header.

Hummen et al. proposed a scheme using the DTLS session resumption [11]. Using a handshake delegator by transmitting session state to a constrained node, the constrained device can reduce the handshake process. Through the session state, the device and service can implement session resumption. This measure induces an advantage in the fact that the authentication certificate does not need to be stored in devices with limited resources. However, a device playing the role of server needs to manage the session state. If a constrained device acts as a server, the constrained device has to keep the session state, which can be a burden on a constrained device.

The IETF CoRE WG proposed TLS_PSK_WITH_AES_128_CCM_8 as a basic cipher suite of DTLS to reduce problems like packet fragmentation and loss and delay in an LLN. However, there is a limitation in the fact that the devices cannot use this cipher suite without a preshared key (PSK). To address this, Gerdes et al. proposed a system in which a ticket is issued [13]. After conducting a DTLS handshake between delegators, each delegator generates a ticket. The DTLS handshake is carried out by the CoAP server and the CoAP client using the ticket. In the ticket, a preshared key is included. Through this, key distribution is made possible to form a DTLS channel based on preshared keys between nodes for which security policy has not been determined in advance. However, another system proposed in [14] focused on the key exchange between devices where a PSK is not exchanged. Thus, the problems caused by DTLS handshakes between CoAP devices within an LLN still remain. Table 1 compares our proposed scheme to related work in two major aspects (i.e., aspects of LLN and constrained node).

3. Prerequisites

3.1. Assumptions. We assume that a constrained CoAP sensor (denoted by CCS in our proposed system) is a highly constrained device. Namely, we mainly consider the devices categorized in Class 1 (Class 0 might be possible) specified in [3], in which the devices in Class 1 have 10 KB or less of RAM and 100 KB or less of ROM. The devices included in Class 0 have more limited resources than the devices in Class 1. The devices in Classes 0 and 1 are assumed to have only source codes corresponding to the Record protocol of DTLS (i.e., an AES module for encryption and a Hash module for MAC creation).

3.2. Conceptual System Architecture. In this section, we propose an efficient and secure system based on standard protocols, especially the use of CoAP/DTLS/UDP in RESTful Web environments [5]. The core component of the proposed system is the Secure Service Manager (SSM) which consists of a host server (e.g., Web server), a resource directory (RD) server, a secure bootstrapping controller including an authentication server, and a DTLS handshake delegator. Each component of the SSM can be implemented as a separate system depending on the IoT architectural environment (e.g., scale), or they can be integrated into a single system. Furthermore, the SSM can be implemented as a virtual system in cloud computing. A CoAP device has characteristics that can take the role of both server and client in the IoT.

Figure 2 shows a conceptual architecture of the proposed system. Parts A and B denoted in Figure 2 describe two different resource-constrained environments. Part A is a static environment, such as a home network, and Part B is a moving environment, such as a human-centric healthcare body area network (BAN). In such environments, the SSM can be implemented in a specific set-top box (Part C in Figure 2) or in a smart phone (Part D). The SSM allows resource-constrained devices to establish a secure session between end applications (Part F). Note that a secure session is a logical path, namely, an actual data path from a client to a server established by using a routing protocol (Part E in Figure 2).

| Related work          | Key technique                  | Transmission overhead in LLN | Fragmentation in LLN | Public key cryptography in sensor | Required memory for sensor | Mobile user support |
|-----------------------|--------------------------------|------------------------------|----------------------|-----------------------------------|----------------------------|---------------------|
| CoAP over DTLS [3]    | Standard recommended           | High                         | High                 | Yes                               | High                       | No                  |
| Bergmann et al. [9]   | Lightweight DTLS               | Medium                       | Medium               | No                                | Medium                     | No                  |
| Raza et al. [10]      | Header compression for DTLS    | Low                          | Medium               | Yes                               | High                       | No                  |
| Hummen et al. [11]    | Session resumption             | Low                          | Low                  | No                                | High (for server)          | No                  |
| Proposed scheme       | DTLS phase separation          | Low                          | Low                  | No                                | Low                        | Yes                 |
The roles of the host server, the RD server, the secure bootstrapping controller, and the DTLS delegator constituting a Secure Service Manager (SSM) are described as follows.

3.2.1. Host Server. The host server manages the main page of a domain (CoAP://home.com). When a client requests data using the uniform resource identifier (URI), the main domain page is provided. The main page provides a user interface to access the resource-constrained sensor and shows the data from the sensor. The host server manages the latest information on the registered sensors and shows it to the user using an RD server and an RD lookup [14] in advance.

3.2.2. Resource Directory Server. This server manages sensor information within the same domain. In the proposed system, information on whether to delegate the DTLS handshake process is stored in the RD table. The information below should be stored in the resource directory parameters table described by Shelby et al. [14]:

Name: DTLS handshake delegation
Query: dg
Validity: Boolean
Descriptions: DTLS handshake delegation for constrained device.

3.2.3. DTLS Handshake Delegator. The DTLS delegator carries out the DTLS handshake process to solve problems caused by the use of DTLS on constrained devices within an LLN. If the CCS is a server, a DTLS connection is created between the DTLS delegator and the CoAP client using the Web browser. If the CCS is a client, the DTLS connection is created between the DTLS delegator and the Web server. If the DTLS handshake process ends, the delegator sends the session ID, DTLS version, and key material created in the handshake process to a CCS. After sending them, the delegator does not interfere with encryption or decryption of application data. The delegator stores and manages session ID, the name of the CCS, and the CCS’s IP address to relay an encrypted message, as shown in Table 2.

3.2.4. Secure Bootstrapping Module. All sensors of the proposed scheme perform the registration process in advance. A preshared key (PSK) between the DTLS handshake delegator and the CoAP sensor is initially configured by secure bootstrapping module. The module includes secure bootstrapping controller and authentication server. To set up the PSK in the sensor, secure bootstrapping must be done (see Section 4 for details).

### Table 2: Relay reference table.

| Session ID | Name of the sensor | IP address of the sensor |
|------------|--------------------|-------------------------|
| EFER235FE, ..., EA | CCS (N/A) | 203.252.1.2 |
| C32AAC2, ..., 3C | CCS_B (bulb) | 203.252.1.3 |
| FABC2123E, ..., FF | CCS_E (energy) | 203.252.1.4 |

4. Secure Bootstrapping

Secure bootstrapping is regarded as a difficult problem in the IoT due to the fact that lots of connected things, such as sensors and actuators, are resource-constrained. In particular, the user-device interfaces that they have are not good enough to do configurations by hand (i.e., inadequate or even without input/output equipment like a display or keyboard). Also, in general cases, small connected things are manufactured and installed by different entities [15]. In such cases, things are initially configured by an installer or a manufacturer during an enrolment phase or a manufacturing configuration phase.
If a secure credential, including a PSK, must be configured in this phase, trust between installer (or manufacturer) and system administrator is extremely important. However, this is not an easy process because manufacturers, installers, and service providers do not share tight and trusting relationships in most cases.

As a solution, this section proposes a secure bootstrapping method that allows a connected thing to reconfigure a symmetric key (i.e., a preinstalled key from an enrolment phase) automatically upon joining an existing network. The basic idea is inspired by key combination (i.e., password) locks on a suitcase. A simple and default password, such as “0000” or “1234,” is initially set up on a suitcase lock when selling it. A new owner can change the password after the purchase. Similarly, in our method the initial key of a node is configured by the installer during an enrolment phase. When the node joins an existing network, the PSK can be securely reconfigured.

We do not assume that a system administrator trusts an installer (or manufacturer) even though he issues orders for the installer. In an enrolment phase, the following transactions should be made.

1. The system administrator makes orders and requests the initial setup of devices to an installer. Presetup information is a set of values including node ID and network ID (i.e., IP address and port number) of the controller (C) for each of the devices and a temporary key used as an initial key.
2. The system administrator also stores the same initial information for each of the nodes in the authentication server (AS).
3. The installer purchases devices and then configures the information requested by the administrator during the installation phase (a part of the enrolment phase). Some of the information for a node may be preconfigured by the manufacturer.
4. When a node joins a network, it knows the network ID of its associated controller, with which it can communicate. Also, the AS has lists including node IDs and preinstalled keys for new nodes.
5. The PSK reconfiguration phase can then be started.

There are three message exchanges between a newly joined node and network nodes (i.e., C and AS). A single system may include functions of controller, authentication server, and DTLS handshake delegator, depending on the size of the application domain or the ability of the controller. In the paper, we assume that the controller and DTLS handshake delegator are implemented in the same device such as user’s mobile phone or set-top box in home. Mutual authentication and PSK reconfiguration procedures are shown in Figure 3.

When a new node joins an existing network, the node generates a random number (RN_d) and sends the RN_D with the node’s identifier (ID_d) to the controller. The network ID of the controller (i.e., IP address and port number) has been preconfigured by the installer for the first connection.

Upon receiving the message, the controller generates a random number (RN_c), time stamp (TS), and a sequence number used as a transaction ID (TID). Then he sends the three values with his own identifier (ID_c) and the message received from the new node to the authentication server. The TS allows the authentication server to derive the valid time of the key and verify the freshness of the arrived message. The TID is used to distinguish a transaction for matching response.

The authentication server first finds the initially configured key material for node D (i.e., IK_d) in its secure repository. The AS can now derive a new PSK for node D and replace IK_d with the PSK, where the PSK for node D is derived as follows:

\[
PSK_d = E(RN_d, V, RN_c, IK_d),
\]

where V is the exclusive-OR operation.

After the reconfiguration of the PSK for node D, the AS encrypts the concatenation value of IK_d, ID_d, and TID with the symmetric key, SK_cs, which is a shared key between the controller and the authentication server. This is because the controller does not have IK_d at that moment. The authentication server then sends the encrypted value to the controller.

On receiving the encrypted value from AS, the controller can know IK_d, thereby calculating the PSK. The controller encrypts the concatenation value of RN_d and RN_c with IK_d. Then, he sends both the encrypted value and his own ID_c to the new node D.

The new device can verify the authenticity of the controller by using the decrypted RN_d value from the received message. Finally, the controller can configure its PSK, thereafter sending the encryption value of RN_c with the PSK to the controller for authentication. The DTLS handshake delegator can then use the PSK. According to a full bootstrapping policy, the PSK can be used for mechanisms of session key derivation. As discussed by Bersani and Tschofenig [16], it goes without saying that a single cryptographic primitive may not support extensible security services, such as identity protection and perfect forward secrecy. However, small devices constituting the IoT might not inherently support all security services. Service developers should therefore strictly define a scope for services and consider a trade-off between capability and security.

5. DTLS Session Establishment

A CoAP sensor can be a client or server depending on the service provided. In this section, we present our proposed system under two scenarios. One is inbound. In the inbound case, a DTLS handshake request is sent to the CoAP sensor in the LLN from outside the LLN (i.e., from the Internet). In the other scenario, the CoAP sensor requests a DTLS handshake from inside the LLN (i.e., it is outbound).

5.1. Inbound Session Establishment. Figure 4 shows the message flow in the process where the CoAP client is sending a request to the CCS in the LLN using a Web browser.

The CoAP client requests the information page of the sensors managed by the “http://www.home.com/” host using the
A new device (D) initiates the secure bootstrapping process by communicating with a controller (C) and an AuthN server (AS). The process involves the following steps:

1. The new device (D) sends an ID and RN to the controller (C).
2. The controller (C) sends an ID, ID, RN, RN, RN, TS, TID, and SK to the AuthN server (AS).
3. The AuthN server (AS) responds with E(RN, PSK).
4. The controller (C) sends ID, ID, RN, RN, RN, TS, TID, and SK to the new device (D).
5. The new device (D) sends E(RN, SK).

Figure 3: Secure bootstrapping process flow.

In the inbound scenario, the message flow begins with a main page request from the web browser to the host server (CoAP://home.com). The host server sends the main page containing the information on the registered sensors to the CoAP client by referring to the resource directory. Then, the CoAP client requests the CSS information using a main page link. The response message takes the following form:

\[ \text{Res: 3.02 Found } \{\text{CoAP://IP:port}/CSS\}. \]  \hspace{1cm} (2)

The host server sends a DTLS handshake notification message to the DTLS delegator and the CSS in line with the CoAP protocol message format. The CoAP client receiving the redirected message sends a DTLS handshake request message to the DTLS delegator and carries out the handshake process. After successfully creating a DTLS session, the DTLS delegator sends the session ID, chosen cipher suite, DTLS

Figure 5 demonstrates the host server’s request asking for information on whether to use the CSS’s delegator. If the CSS uses the delegator, the IP address and port of the delegator are sent.

For secure communications between CSS and a CoAP client, the host server sends a redirected message containing the delegator IP address and port information of the CSS. The response message takes the following form:
version, and key material to the CCS by encrypting it with \textit{PSK.DN}. The message format is as follows:

\[
E(\text{Session ID} \parallel \text{PSK.DN})
\]

The CCS sends the session ID encrypted with \textit{PSK.DN} and the encrypted data together with a DTLS encryption key from the key material in line with the Record protocol format:

\[
E\left(\begin{array}{l}
\text{Session ID}, \text{PSK.DN} \\
\text{data}, \text{DTLS enc key}
\end{array}\right)
\]

The delegator relays the message, \(E(\text{data}, \text{DTLS enc key})\), to the CoAP client by referring to Table 2 and by decrypting \(E(\text{Session ID, PSK.DN})\).

5.2. Outbound Session Establishment. If the CCS sends data to a Web server on the Internet, communications can be conducted more simply than in the inbound scenario (see Figure 6). As we mentioned in Section 5.1, the CCS assumes that there is a key, \textit{PSK.DN}, safely shared in advance with the handshake delegator.

The CCS requests handshake delegation from the handshake delegator in line with the CoAP message format. The delegator that receives the request from the CCS creates a DTLS handshake session with the Web server. When the DTLS session is successfully open, the session ID and content type, version information, and key material (encrypted with \textit{PSK.DN}) are sent. The delegator stores the information concerned in the table, using Table 2 to relay the message. The CCS encrypts the data with the DTLS encryption key from the key material in line with the DTLS record layer format by decrypting the transmitted message. Then, the CCS encrypts the session ID with \textit{PSK.DN} and sends the data together to the delegator. The delegator relays the message using the session ID in the relay reference table. The process mentioned here is the same as communication between delegator and CCS for inbound communication.

To establish a secure session between resource-constrained nodes, “inbound and outbound establishment schemes” can be used together. If two CoAP sensors have their DTLS delegator, they can establish a secure session by using the proposed system. When a client sensor needs to establish a secure session toward a server sensor, the client’s delegator sends DTLS handshake request message. This is the same as the outbound scenario from the aspect of the client sensor. After finishing the DTLS handshake phase between client’s delegator and server’s delegator, session information is encrypted and sent to each of the constrained sensors. From the aspect of the server sensor, this flow is the same as the inbound scenario.

6. Implementation and Evaluation

We implemented the proposed scheme based on TinyDTLS (http://tinydtls.sourceforge.net/), which is a lightweight DTLS library. The old version of TinyDTLS does not support public key cryptography (i.e., it supports DTLS with PSK only), but the ECDSA was implemented in a recent version (v.0.8.1). We registered callback functions to \texttt{dtls.set_handle()} for both the DTLS server (resource-constrained node located in LLN) and the SSM (especially for delegator) as shown in Figure 7. Major modifications of the TinyDTLS library to implement SSM are as follows:

(i) removing Static Keyword of some variables to allow application code of the library to access functions for adding/deleting peer information;

(ii) implementing module to allow SSM to send “\texttt{Key Block}” to entity that only has record layer of TinyDTLS;

(iii) implementing handler of callback function to send “\texttt{Key Block}” and to relay encrypted data to thing.

The handshake part of the DTLS library has been removed for a resource-constrained node in the proposed scheme. The processing of handshake in DTLS requires lots of transmission overhead and computing power. The size of DTLS handshake message is very large to deliver in LLN (according to the specification, in theory up to \(2^{24} - 1\) bytes, in practice many kilobytes [7]). Therefore, functionality of data fragmentation is necessary in either IP layer or DTLS layer to deliver such a big message over wireless link configured with small size of MTU. The less fragmentation is done in
LLN, the more performance is enhanced, and the stronger security can be achieved as well [4, 17]. To compensate for the handshakeless DTLS library, the node should set “Key_Block” to “security_params” of a forcibly assigned session and then initialize the epoch and sequence number. Figure 8 illustrates the processing flow of each component of the proposed system and a screen shot of each of the components.

Figure 9 shows the screen shots for the client and the server in detail. Note that SSM including delegator does not decrypt the data from the client to the server but merely relays the data.

In addition, we developed a test program based on the Contiki OS [18] and TinyDTLS to measure power consumption of each of the flights required for DTLS session establishment. Computing time is also tightly related to the power consumption. In addition, we utilized the Cooja simulator because it is difficult to measure the power consumption in a real test-bed due to various reasons, such as signal interference and the state of the charge of the battery. The Contiki OS

---

### Figure 7: SSM server implementation.

| SMM server application |
|-------------------------|
| `ssm_server.c`          |
| - read_from_peer()      |
| - send_to_peer()        |
| - relay_to_server()     |
| - get_psk_info()        |
| - create_socket()       |

### Figure 8: Processing overview.

- **Web client** (Internet node)
- **SSM** (DTLS HS delegator)
- **LLN server** (record layer)

DTLS handshake

Send encrypted Key_Block

Application message (cipher-text)

End entity to encrypt plaintext

Relay to server

End entity to decrypt ciphertext

---

```
int main() {
    int fd = socket();
    dtls_init();
    dtls_set_handler();
    dtls_new_context(&fd);
    while (1) {
        //dtls_handle_message();
        //or dtls_write();
    }
}
```
includes MSPSim, which is a simulator for the MSP430 chipset, and Cooja, which is a graphical user interface (GUI) for the simulator. The simulation environment is summarized in Table 3.

Like [17], we also used the energy measuring method provided by the Contiki OS; in particular we used the energest() function in the Contiki OS to measure the power consumption in Cooja. The function gives us the CPU usage time, LPM (reduced power CPU), and transmit/listen power consumptions. It was reported that the energest() function has 94% of accuracy compared to the real world [19]. Cooja supports the standard output window as shown in Figure 10 for monitoring the measured value.

Figure 11 shows that the power consumption required for DTLS handshake processing (total 26,028 mA) is more than nine times higher than the consumption of record layer processing (2809 mA).

In particular, the increasing loss rate of an LLN leads to a rapid increase in power consumption and computing time because of retransmission. It was reported that the amount of energy consumption for transmission and reception of frame over wireless link is about 10 times higher than the computing [17]. In addition, the small MTU size of an LLN (e.g., 127 bytes under IEEE 802.15.4) requires data fragmentation in flights 5 and 6. Such fragmentation in an LLN makes for a worse situation with respect to performance and security as well (see Section 7 in detail). In the proposed system, the handshake layer of DTLS is removed from a resource-constrained node so that less fragmentation occurs in the LLN (actually, almost zero fragmentation in doing handshake phase), thereby resulting in more efficiency and security.

### 7. Security Analysis

#### 7.1. SSM Spoofing

If an attacker pretends to be the SSM, the attacker can get all the information about the session caused by delegating the DTLS handshake. So, encrypted data between end nodes can be revealed to the attacker. In our
proposed system, PSK (which is shared between the SSM and a constrained device in the secure bootstrapping phase) is used to deal with this attack. Because data is encrypted by using PSK and then sent, the attacker cannot deceive a constrained device and cannot access the encrypted data.

7.2. Semi-End-to-End Security. The SSM can get all session information by delegating the DTLS handshake. It can be an end-to-end security matter. But encrypted session information is sent to a constrained device immediately, and the SSM does not store session information. Thus, end nodes participating in the DTLS communication will encrypt and decrypt data themselves. The SSM performs only simple data relay after transmitting the session information to the constrained device. In our system, the performer of the encryption and decryption is the end node in the DTLS communication. We note here that, in particular, the SSM should be a trusted and preregistered device, such as a smart phone of the user. Therefore, end-to-end security (semi-e2e security, exactly) is guaranteed.

7.3. Denial of Service. The devices constituting the IoT have low CPU performance and a small amount of memory. Sending a DTLS handshake request to these devices can appear to be a DoS attack, even though the request is from a legitimate user. If an attacker sends a DTLS handshake message to a constrained device directly, conditions in the LLN and the device become much worse. An SSM helps to solve the DoS problem by delegating the DTLS handshake. The SSM prevents constrained devices from directly receiving a lot of messages.

7.4. Single Point of Failure. Several approaches utilizing delegation can lead to a single point of failure (SPOF), which is one of the most conventional but critical problems in the security field. In our scheme, the SSM has an important role in delegating the DTLS handshake instead of several CoAP sensors. Thus, if the SSM is compromised or fails, then none of the sensors supported by the SSM can create a secure session with a client or server outside the LLN.

A well-defined trust manager can mitigate such a SPOF problem. The trust manager can select another authenticated device as a new SSM; it then broadcasts related information to its sensors. The newly selected SSM should be a resource-rich device in a smart home or smart building (e.g., a smart refrigerator, set-top box, etc.). Alternatively, a virtually implemented SSM in a cloud system can also be a proper solution. It is more difficult to compromise a virtual SSM in cloud computing, which is managed and monitored by a security administrator, than to attack a home device or smart phone, which is managed by a regular user.

We note here again that there is a secure registration method between the SSM and IoT devices controlled by the SSM, as discussed in Section 3. Moreover, the fundamental assumption of the paper is that the secret key shared by both SSM and its devices cannot be compromised. In further work, we intend to design and implement a concrete secure system with additional mechanisms including secure bootstrapping, trust management, and key revocation.

7.5. Fragmentation Attacks. The different MTU sizes between the Internet and an LLN require a packet fragmentation mechanism. 6LowPAN, which is an IPv6 adaptation layer, supports a method to fragment a large IPv6 packet into multiple small frames. Generally, sensing data and control data for actuators might be small. However, the size of a DTLS handshake message is bigger than the maximum frame size of an LLN, as in IEEE 802.15.4 (i.e., 127 bytes). In particular, DTLS fragmentation is inevitable at the fourth flight of a DTLS handshake, which contains a relatively large size certificate for the server and key exchange message. According to the description by Park and Kang [8], 27 DTLS fragmented datagrams should be transferred when using “TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8” with a raw public key certificate. These fragmented datagrams lead to significant transmission overhead due to the header added to each of the frames.

An even worse problem is that the lack of an authentication mechanism at the 6LowPAN layer allows attackers to launch fragmentation attacks, fragment duplication attacks, and buffer reservation attacks presented by Hummen et al. [4]. An attacker eavesdrops and modifies a fragmented frame in the middle of the wireless multihop link to launch the fragment duplication attack. The target node cannot figure out the modified frame at the time of reception. As a result, just a single forged frame sent by the attacker can prevent successful reassembly in the target node. Moreover, the target node must discard all frames in the buffer and wait for retransmission, thereby resulting in a DoS attack. The buffer reservation attack utilizes the fact that the first frame reserves a memory space for reassembling the original packet as indicated in the header (i.e., the datagram size field) at the target node. The attack is simply launched by sending a forged start frame containing a large number in the datagram size field.

The fragmentation issue in an LLN is the major motivation for the work in this paper. The proposed scheme almost removes fragmentation in an LLN thanks to the delegator. Therefore, our scheme can mitigate fragmentation attacks in LLNs. Also, in terms of network efficiency, we reduce overhead caused by message fragmentation, thereby reducing frame loss rate and delay in LLNs.

8. Conclusion

We proposed an efficient scheme using a technique for delegating the DTLS handshake phase. From the perspective of a constrained network like an LLN, network transmission overhead, delay, and loss problems caused by fragmented handshake message packets are solved by delegating the handshake. From the perspective of a constrained device, the device does not have to store fragmented handshake packets in the buffer until receiving all of them. Besides, a constrained device can use DTLS communication without any source code for a DTLS handshake.

Although we separate the DTLS protocol into a handshake phase and an encryption phase, end-to-end security is guaranteed because data encryption and decryption are done in the end node. And this system can deal with an SSM spoofing attack and DoS attacks on a constrained device. The
SSM and the constrained device are physically separate but can virtually be regarded as a single system in a trusted relation via a preshared key. So, our system can help to deploy constrained devices in a secure manner in constrained environments.

**Conflict of Interests**

The authors declare that there is no conflict of interests regarding the publication of this paper.

**Acknowledgments**

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2015-H8501-15-1008) supervised by the IITP (Institute for Information & Communications Technology Promotion). Also, this research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (no. 2014R1A1A2056961).

**References**

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of things (IoT): a vision, architectural elements, and future directions,” *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] J. Bradly, J. Barbier, and D. Handler, “Embracing the internet of everything to capture your share of $14.4 Trillion,” White Paper, Cisco, 2013.

[3] C. Bormann, M. Ersue, and A. Keranen, “Terminology for constrained-node networks,” RFC 7228, Internet Engineering Task Force (IETF), 2014.

[4] R. Hummen, J. Hiller, H. Wirtz, M. Henze, H. Shafagh, and K. Wehrle, “6LoWPAN fragmentation attacks and mitigation mechanisms,” in *Proceedings of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec ’13)*, pp. 55–66, ACM, April 2013.

[5] Z. Shelby, K. Hartke, and C. Bormann, “The constrained application protocol (CoAP),” RFC 7252, IETF, 2014.

[6] K. Hartke, “Practical issues with datagram transport layer security in constrained environments,” Internet Draft, IETF, 2014.

[7] S. Raza, H. Shafagh, and O. Dupont, “Compression of Record and Handshake Headers for Constrained Environments,” IETF, http://tools.ietf.org/html/draft-rama-dice-compressed-dtls-00.

[8] R. Hummen, J. H. Ziegeldorf, H. Shafagh, S. Raza, and K. Wehrle, “Towards viable certificate-based authentication for the Internet of Things,” in *Proceedings of the 2nd ACM Workshop on Hot Topics on Wireless Network Security and Privacy (HotWiSec ’13)*, pp. 37–41, Budapest, Hungary, April 2013.