Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 130 (2018) 432–439

Procedia Computer Science

www.elsevier.com/locate/procedia

The 9th International Conference on Ambient Systems, Networks and Technologies (ANT-2018)

Performance Analysis of End-to-End Security Schemes in Healthcare IoT

Sanaz Rahimi Moosavi*, Ethiopia Nigussie, Marco Levorato, Seppo Virtanen, Jouni Isoaho

Department of Future Technologies, University of Turku, 20014 Turku, Finland

Abstract

In this paper, we analyze the performance of the state-of-the-art end-to-end security schemes in healthcare Internet of Things (IoT) systems. We identify that the essential requirements of robust security solutions for healthcare IoT systems comprise of (i) low-latency secure key generation approach using patients' Electrocardiogram (ECG) signals, (ii) secure and efficient authentication and authorization for healthcare IoT devices based on the certificate-based datagram Transport Layer Security (DTLS), and (iii) robust and secure mobility-enabled end-to-end communication based on DTLS session resumption. The performance of the state-of-the-art security solutions including our end-to-end security scheme is tested by developing a prototype healthcare IoT system. The prototype is built of a Pandaboard, a TI SmartRF06 board and WiSMotes. The Pandaboard along with the CC2538 module acts as a smart gateway and the WisMotes act as medical sensor nodes. Based on the analysis, we found out that our solution has the most extensive set of performance features in comparison to related approaches found in the literature. The performance evaluation results show that compared to the existing approaches, the cryptographic key generation approach proposed in our end-to-end security scheme is on average 1.8 times faster than existing key generation approaches while being more energy-efficient. In addition, the scheme reduces the communication overhead by 26% and the communication latency between smart gateways and end users by 16%. Our scheme is also approximately 97% faster than certificate based and 10% faster that symmetric key-based DTLS. Certificate based DTLS requires about 2.9 times more ROM and 2.2 times more RAM resources. On the other hand, the ROM and RAM requirements of our scheme are almost as low as in symmetric key-based DTLS.

Keywords: Smart Home/Hospital; Cryptographic Key Generation; Bio-Electrical Signal; Authentication and Authorization; End-to-End Security

1. Introduction

IoT enables physical objects in the physical world as well as virtual environments to interact and exchange information with each other in an autonomous way so as to create smart environments. Healthcare IoT systems are distinct in that they are built to deal directly with the data of human health conditions, which inherently raises the requirements of security, safety and reliability. In addition, they have to offer real-time notifications and responses about the status of patients. In healthcare IoT systems, security and privacy of individuals are among major areas of concern as most devices and their communications are wireless in nature. This is to prevent manipulating and eavesdropping on sensitive medical data or malicious triggering of specific tasks. Key security requirements for healthcare IoT systems consist

[*] Corresponding author. Tel.: +3-582-333-8647.
E-mail address: saramo@utu.fi

10.1016/j.procs.2018.04.064

of three main phases: (1) secure cryptographic key generation, (2) authentication and authorization of each healthcare IoT component, (3) and robust and secure end-to-end communication between sensor nodes and health caregivers are critical requirements[1]. Existing security and protection techniques including cryptographic key generation solutions, secure authentication and authorization, robust end-to-end communication protocols, and privacy assurance cannot be re-used due to the following main reasons: (i) proposed security solutions must be resource-efficient as medical sensor nodes used in healthcare IoT systems have limited memory, processing power, and communication bandwidth, and (ii) medical sensor nodes can be easily abducted or lost since they are tiny in terms of size. To mitigate the above-mentioned risks, robust and lightweight security solutions are needed.

In this paper, we analyze the performance of the state-of-the-art end-to-end security solutions in healthcare IoT systems. The main contributions of this paper are twofold. First, we identify and present the essential requirements of robust security solutions for healthcare IoT systems which include (i) secure ECG-based cryptographic key generation, (ii) authentication and authorization of each healthcare IoT component based on certificate-based Datagram Transport Layer Security (DTLS), and (iii) secure mobility-enabled end-to-end communication based on session resumption technique as well as the concept of fog layer in IoT for realizing efficient and seamless mobility.

The remainder of this paper is organized as follows: Section 2 provides an overview of related work. Section 3 discusses the architecture and requirements of healthcare IoT systems. Section 4 presents our healthcare IoT security solutions. Section 5 provides a comprehensive performance analysis of different security solutions. In this section, the comparison of our work with similar existing approaches is also presented. Finally, Section 6 concludes the paper.

2. Related Work

To establish an efficient inter-operable network security between end-points, variants of end-to-end security protocols have been proposed, among which DTLS is one of the most relevant protocols[2]. DTLS comprises of four main protocols: Handshake, Alert, Change Cipher Spec, and Record. The most recently DTLS-based solutions are proposed by Hummen *et al.*[3], Zack *et al.*[4], Granjal *et al.*,[5] and Kang *et al.*[6]. In[4], authors proposed symmetric key-based DTLS solution as the basic cipher suite of DTLS to reduce packet fragmentation, loss and delay in a low-power and lossy network. However, there is a limitation in the fact that the sensor devices cannot utilize this cipher suite without a pre-shared key (PSK). In[7], authors present a certificate-based raw public key cipher suite. This cipher suite comprises of six flight messages which are fragmented into 27 datagram packets. Nevertheless, packet fragmentation causes issues such as high data loss rate and packet re-transmission delays. To reassemble a fragmented message packet, sensor devices have to keep fragmented pieces of the message in the buffer until all the pieces arrive. This is a considerable burden to the resource-constrained sensor devices. In other works presented in[3,5,6], the authors present an implementation of delegation-based architecture which relies on a delegation server/certificate authority. Their solutions, however, lack scalability and architecture reliability as their proposed architectures are based on a centralized delegation server/certificate authority or on the centralized 6LoWPAN Borader Router (6LBR). The main reason is that their proposed architectures cannot be extended to be utilized in multi-domain infrastructures, such as large hospital environments. If a malicious adversary performs a DoS attack or compromises the 6LBR, a large quantity of stored information concerning the constrained domain can be retrieved. These issues are solved in our scheme as the architecture is distributed. To be more specific, in our scheme, in a multi-domain smart home/hospital environment, if an attacker runs a DoS attack or compromises one of the smart gateways, only the associated medical sub-domain is disrupted. We believe that the approaches presented by Granjal *et al.*[5] and Kang *et al.*[6] do not provide comprehensive end-to-end security. Rather, they can be considered *semi end-to-end* security. This is beacuse in these works, the 6LBR acts as an intermediary node located between the sensor and the end-user. Every time these two end-points try to communicate with each other, all the secret information related to the communication needs to pass through the 6LBR. Whilst, the smart gateway utilized in our work is only used during the initialization phase, and then afterwards, both end-points directly communicate with each other through a channel secured by the DTLS session resumption. Although Hummen *et al.*s'[3] proposed delegation-based architecture offers end-to-end security, it is still not secure against the DoS attack due to the use of a centralized delegation server. Their presented architecture also suffers from shortcomings in architecture reliability and scalability which is mainly due to the reasons mentioned above.

3. Healthcare IoT: Architecture and Requirements

In a typical healthcare IoT system, to monitor patients' vital signs and activities, the system has to ensure the security and privacy of patients. Physicians and other caregivers demand a dependable system in which the results are accurate, timely and the service is reliable and secure. To guarantee these requirements, the smart components in the system require a predictable latency, reliable and robust communication with other components of healthcare IoT systems[8]. The 3-layer system architecture of our proposed healthcare IoT system on which the security solutions can be applied is shown in Figure 1. In such a system, patients' health-related information is recorded by wearable or implantable medical sensor nodes with which the patient is equipped for personal monitoring of multiple parameters. The functionality of each layer is as follows: (1) *Device Layer*, the lowest layer consisting of several physical devices

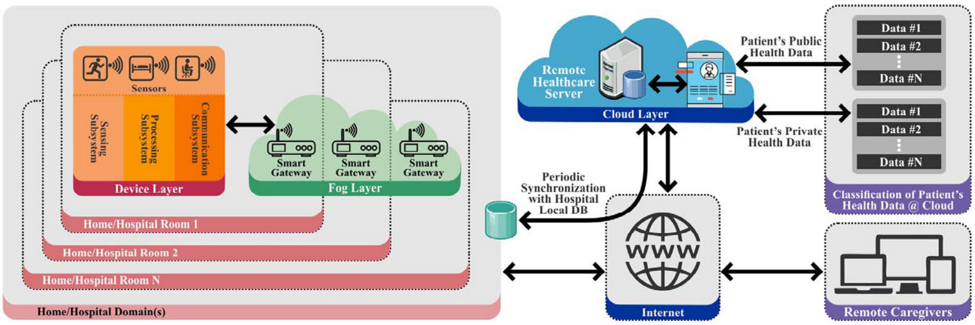

Fig. 1: The system architecture of our healthcare IoT system with secure end-to-end communication

including implantable or wearable medical sensor nodes that are integrated into a tiny wireless module to collect contextual and medical data. (2) *Fog Layer*, the middle layer consists of a network of interconnected smart gateways. A smart gateway receives data from different sub-networks, performs protocol conversion, and provides other higher level services. It acts as repository (local database) to temporarily store sensors' and users' information, and provides intelligence at the edge of the network. (3) *Cloud Layer*, the cloud layer includes broadcasting, data warehousing and big data analysis servers, and a hospital local database that periodically performs data synchronization with the remote healthcare database server in the cloud.

4. Healthcare IoT Security Solutions

As we comprehensively discussed in[1], key security requirements for healthcare IoT systems consist of three main phases: (i) secure and efficient cryptographic key generation for healthcare IoT devices, (ii) authentication and authorization of each healthcare IoT component, and (iii) and robust and secure end-to-end communication between medical sensor nodes and health caregivers. In the following, we briefly present our healthcare IoT security solutions.

4.1. ECG Feature-Based Cryptographic Key Generation

Since medical sensor nodes deal with patients' vital health data, securing their communication is an absolute necessity. Without robust security features not only patients' privacy can be breached but also adversaries can potentially manipulate actual health data resulting in inaccurate diagnosis and treatment. Medical sensor nodes rely on cryptography to secure their communications[9]. Proper application of cryptography requires the use of secure keys and robust key generation methods. Key generation approaches that are proposed for wireless networks in general are not directly applicable to tiny medical sensors as they are highly resource-constrained and demand a higher security level. Given the constrained nature of medical sensor nodes used in healthcare Iot systems, conventional key generation approaches may potentially involve reasonable computations as well as latency during network or any subsequent adjustments, due to their need for pre-deployment. In[10], we presented two different ECG-based cryptographic key generation approaches. The first approach is integrating interpulse interval (IPI) sequence of ECG signal with pseudorandom number that is generated using Fibonacci linear feedback shift register. The generated key is called IPI-PRNG. An alternative key generation approach that utilized the Advanced Encryption Standard (AES) algorithm and IPI sequences as the seed generator for the AES, called IPI-AES. IPI-PRNG and IPI-AES offer higher security levels compared to conventional key generation approaches. In[11], we further improved the ECG-based key generation approach by introducing the use of several ECG Features (SEF) that reduce the key generation execution time overhead significantly while preserving the achieved high security levels. The proposed approach is applied to both normal and abnormal ECG signals. The SEF approach uses 4 main reference-free [1] features of the ECG signal (being extracted from every ECG heartbeat cycle) along with consecutive IPI sequences to generate ECG-based cryptographic keys. To reinforce and enhance the security level of our approach, we consolidate the SEF key generation approach with two different cryptographically secured pseudo random number generators, called, SEF-PRNG and SEF-AES. We evaluated the efficiency of our IPI-PRNG, IPI-AES, SEF, SEF-PRNG, and SEF-AES approaches by simulations on real ECG data from different subjects having various heart health conditions.

4.2. Mutual Authentication and Authorization of Healthcare IoT Components

In the paradigms of healthcare IoT, not only data can be collected by medical sensor nodes and transmitted to end-users, but end-users can also access, control, and manage medical sensors through the Internet. As a result, mutual authentication and authorization of end-users and devices used in healthcare IoT systems is a crucial task. Our proposed architecture, called *SEA*, exploits the role of smart e-health gateways in the fog layer to perform the

[1] In this context, the reference-free property indicates a dynamic technique in which no ECG fiducial point is fixed as reference.

authentication and authorization of remote end-users securely and efficiently on behalf of the medical sensors[12]. *SEA* focuses on a fact that the smart e-health gateway and the remote end-user have sufficient resources to perform various heavy-weight security protocols as well as certificate validation. By providing the established connection context to the medical sensor nodes, these devices no longer need to authenticate and authorize a remote caregiver. It is supposed that within the certificate-based DTLS handshake, from one hand, the smart gateway authenticates the remote end-user through certificates. In this regard, similar to current web browsers, smart gateways hold a pool of trusted certificates. On the other hand, the smart gateway either authenticates to the remote end-user through certificates within the DTLS handshake or based on an application-level password once the handshake is terminated. Once the mutual authentication between the end-user and the smart gateway is done successfully, the end-user authorizes as a trusted entity so that a data query from the end-users' side is transmitted to the medical sensor nodes through the smart gateway. To facilitate the security and authorization of communication, it is required that both entities, the constrained medical sensor node and the smart gateway, also mutually authenticate one another during the initialization phase.

4.3. Secure End-to-End Communication for Mobility Enabled Healthcare IoT

In[1], we enabled secure end-to-end communication between end-points of a healthcare IoT system by developing a session resumption-based scheme which offloads the encrypted session states of DTLS towards a non-resource-constrained end-user. The main motivation to employ the DTLS session resumption is to mitigate the overhead on resource-constrained medical sensors. The session resumption technique is an extended form of the DTLS handshake which enables a client/server to continue the communication with a previously established session state without compromising the security properties. The major advantages offered by our scheme compared to the conventional end-to-end security solution can be found in[1]. We applied our proposed session resumption-based end-to-end security scheme for healthcare IoT to the full system architecture shown in Figure 1. Providing patients with the possibility to walk around the hospital wards knowing that the monitoring of their health condition is not interrupted is an essential feature. To achieve a continuous monitoring of patients considering the mobility support, in[1] , we developed self-configuration/handover mechanisms which are capable of handling secure and efficient data transfers among different medical sensor networks. A fog layer-based data handover mechanism is defined as the process of changing or updating the registration of a mobile sensor from its associated base MSN to the visited MSN, for example, when moving across the hospital's wards. Data handover solutions should enable the ubiquity when they need to work autonomously without human intervention. The handover mechanism should also offer medical sensor nodes continuous connectivity, if there exist several gateways in the hospital or nursing/home environments.

Table 1: Execution time comparison of different ECG-based key generation approaches to produce 128-bit cryptographic keys

Approach	Execution Time Single Iteration (ms)	Execution Time Total (s)	Energy Consumption Single Iteration (μJ)	Energy Consumption Total (mJ)
IPI[9,13]	181.3	2.9	9507.1	527.6
IPI-PRNG	198.6	3.2	11022.3	611.7
IPI-AES	244	3.9	13542	751.5
SEF	104.3	0.9	5788.6	321.2
SEF-PRNG	136.9	1.1	7598	421.6
SEF-AES	168.1	1.3	9884.5	548.5

5. Implementation and Performance Analysis

The system architecture illustrated in Figure 1 is implemented for experimental evaluation for two different scenarios: in-home and hospital room(s). To Implement the proposed healthcare IoT system architecture, we setup a platform that consists of medical sensor nodes, UT-GATE smart e-health gateways, a remote server, and end-users. UT-GATE is constructed from the combination of a Pandaboard and a Texas Instruments (TI) SmartRF06 board that is integrated with a CC2538 module[14]. In our configuration, UT-GATE uses 8GB of external memory and is powered by Ubuntu OS which allows to control devices and services such as local storage and notification. To investigate the feasibility of our proposed architecture, the *Wismote*[15] platform, which is a common resource-limited sensor nodes, is utilized in Contiki's network simulation tool Cooja[3]. For the evaluation, we use the open source tool *OpenSSL* version 1.0.1.j to create elliptic curve public and private keys from the NIST P-256 and X.509 certificates. The server association to the end-user is created using OpenSSL API which provides all necessary functions related to end-users including configuration, certificate, handshake, session state, and cipher suites to support session resumption. *Tiny-DTLS*[16] is used as the code-base of the proposed scheme. For the public-key functions, we utilize the *Relic-toolkit*[17] that is an open source cryptography library tailored for specific security levels with emphasis on efficiency and flexibility. The MySQL database is set up for static and non-static records. The cloud server database is processed using xSQL Lite which is the third party tool for data synchronization. With respect to the cryptographic primitives and to make a fair comparison, we followed similar cipher suites as employed in the most recently proposed authentication and authorization architecture for IP-based IoT[17]. In this regard, we utilize elliptic curve NIST-256 for public-key operations, $AES\_128\_CCM\_8$ (with an IV of 8 bytes) for symmetric-key, and SHA256 for hashing operations. To asses the performance of different ECG-based cryptographic key generation approaches in terms of execution time, we conduct the experiments on ECG signals of 48 subjects with Arrhythmia obtained from the publicly available database, that is, Physiobank[18]. The recordings are digitized at 360 samples per second with 11-bit resolution over a 10 mV range per patient with 16 bit resolution over a range of 16 mV. We have captured 100 different samples of 5 minute long ECG data for each subject. We have implemented the key generation approaches utilizing MATLAB.

5.1. Cryptographic Key Generation Performance Analysis

In this section, we analyze and compare the performance of different ECG-based cryptographic key generation approaches to produce 128-bit cryptographic keys from the execution time and energy consumption point of views.

5.1.1. Cryptographic Key Generation Execution Time

To investigate the generation execution overhead of our approaches compared to the conventional IPI approach, we have examined the execution time required to generate 128-bit ECG-based cryptography keys. For this purpose, we utilized the *Wismote*[15] platform, which is equipped with a 16MHz MSP430 micro-controller, an IEEE 802.15.4 radio transceiver, 128KB of ROM, 16KB of RAM, and supports 20-bit addressing. Our experiments are carried out on ECG recordings obtained from the MIT-BIH Arrhythmia dataset, sampled at 360 Hz.

Table 1 presents the computed key generation execution times of our IPI-PRNG, IPI-AES, SEF, SEF-PRNG, and SEF-AES approaches as well as the conventional IPI approach. The execution times are presented in both single iteration and total times. Single iteration execution time indicates the time required to produce an *8*-bit binary sequence from one heartbeat cycle. Total execution time means the sum of single iteration execution times until successive iterations of the operations yields the desired result, that is, generates the desired 128-bit ECG-based cryptographic keys. Considering a subject with the ECG heartrate of 60 bpm, the specific MSP430 micro-controller requires about 181 ms, 198 ms and 244 ms execution times per iteration for the IPI, IPI-PRNG, and IPI-AES approaches, respectively. These are the times these three approaches require to produce an 8-bit binary sequence from one ECG heartbeat cycle. To generate 128-bit ECG-based cryptographic keys, it is required for IPI, IPI-PRNG and IPI-AES approaches to compute 16 heartbeat cycles from a subject's ECG signal. The same microcontroller requires about 104.3 ms, 136.9 ms, and 178.1 ms execution times for the SEF, SEF-PRNG, and SEF-AES approaches to produce 16-bits binary sequences from one ECG heartbeat cycle. To generate 128-bit ECG-based cryptographic keys, the SEF, SEF-PRNG and SEF-AES approaches need to compute 8 heartbeat cycles from a subject's ECG signal. As a result, the total key generation execution times of SEF, SEF-PRNG, and SEF-AES approaches are calculated as 104.3 * 8=0.9 (s), 136.9 * 8=1.1 (s), and 168.1 * 8=1.3 (s), respectively, which are considerably lower than their counterparts. The key generation execution times of SEF, SEF-PRNG and SEF-AES are in average 1.8 times times faster than IPI, IPI-PRNG and IPI-AES approaches. This is due to the fact that in IPI, IPI-PRNG and IPI-AES in total 8 bits can be extracted from one ECG heartbeat cycle, while in SEF, SEF-PRNG and SEF-AES approaches in total 16 bits can be extracted from the same heartbeat cycle. Thus, by utilizing additional ECG features, the latency of ECG-based key generation approaches can be significantly reduced. It should be mentioned that, generating these cryptographic keys are performed in an on-demand way and not in every message transaction, for example, once the key is revoked.

5.1.2. Energy Consumption Due to ECG-based Key Generation

To measure the consumed energy of each Wismote sensor node due to key generation, we utilize the following equation: $E = U \times I \times t$ where U represents the supply voltage in Volt (V), I is the current draw of the hardware in milliAmperes (mA) , and t is the key generation execution time in milliseconds (ms). According to the Wismote datasheet that is available in[15], the Wismote sensor node has a current consumption of 18.5 mA and a supply voltage of 3 V. The energy consumption comparison of different ECG-based cryptographic key generation approaches are presented in Table 1. According to the results, SEF, SEF-PRNG and SEF-AES have in average better energy consumption than IPI, IPI-PRNG and IPI-AES approaches. This is due the fact that SEF, SEF-PRNG and SEF-AES approaches require lower execution time. Hence, the energy consumption of the Wismote sensor nodes can be considerably reduced.

5.2. Mutual Authentication and Authorization Performance Analysis

In this section, we analyze the performance of different mutual authentication and authorization approaches from the transmission overhead and latency points of views.

5.2.1. Transmission Overhead

The required number of packet fragments has a direct impact on energy consumption of the healthcare IoT devices. In the following, we analyze the transmission overhead in more detail. As we presented in[10], to perform the certificate-based DTLS handshake, all 15 messages are needed to establish a DTLS connection. When transmitted over size-constrained IEEE 802.15.4 radio links, these messages must additionally be split into several packet fragments due to their extensive message size[3]. As Table 2 presents, we compared the transmission overhead of the proposed SEA approach to the most recent architecture for a successful certificate-based DTLS connection. In delegation-based

Table 2: Performance comparison with the most recently proposed authentication and authorization approach for IoT

	SEA Approach (This Work)	Hummen *et al.*[3]	SEA Approach Improvements (%)
Transmission-overhead (byte)	1190	1609	26
6LoWPAN Fragments (#)	18	24	26
Latency-GE (s)	~ 15	~ 15	0
Latency-NG (s)	5.001	6.08	5
Latency-NE (Total) (s)	20.001	21.08	5

architecture, the measured transmission overhead of the certificate-based DTLS handshake is 1609 bytes which cause in total 24 fragments for the transmission of all handshake messages from the delegation server to the end-user[3]. In contrast, our purposed architecture requires transmission of 1190 bytes and it cause 18 fragments totally. As a result, the transmission overhead in our architecture reduces by 26% compared to the delegation-based architecture.

5.2.2. *Authentication and Authorization Latency*

Latency in this context is defined as the time required from sending a request to confirming the shared session key between two peers. To estimate the authentication and authorization latency, the processing time which is spent from sensor node to the end-user, that is, NE is calculated. This processing time is deduced from the summation of communication latency from sensor node to smart gateway, that is, NG and smart gateway to end-user which can be written as: $Latency_{NE}(s) = Latency_{NG}(s) + Latency_{GE}(s)$. To compute the communication latency from the UT-Gate to the end-user, a proxy server is adjoined to the network. The proposed SEA approach achieves an almost equivalent NG processing time to the delegation-based architecture[3], which takes up to 15 *s* for the certificate-based DTLS. However, the proposed SEA approach considerably reduces the processing time required for GE compared to the delegation-based architecture. As shown in Table 2, in SEA, the processing time required for GE is about 5.001 *s* whereas this time increases to about 6.08 *s* in the delegation-based architecture. Regarding the latency from the gateway to the end-user, the proposed SEA architecture obtains about 16% improvement compared to the delegation-based architecture. When utilizing public keys, the certificate-related processing overhead is no longer available. This is a remarkable advantage as the certificate-related overhead increases linearly with the depth of certificate hierarchy.

5.3. *End-to-End Communication Performance Analysis*

We analyze the performance of different end-to-end security schemes for mobility enabled healthcare IoT from (i) sensor-side processing time, (ii) sensor-side energy consumption, (iii) data handover latency between gateways, (iv) client-side processing time, (v) client-side run-time performance, and (vi) memory footprint point of views.

5.3.1. *Sensor-side Processing Time*

The total sensor-side processing time and energy consumption of different DTLS modes to provide end-to-end security is presented in Table 3. For the evaluation, in Cooja, we configured two Wismotes as a client and a server. When the booting process is performed, the client initiates the handshake by sending the *ClientHello* message. After a successful handshake, we measured the total processing time at the sensor-side. Results demonstrated that the symmetric key-based DTLS mode[4] and our session resumption-based scheme require almost similar processing time.

The proposed scheme requires 20 ms less processing time than the symmetric key-based mode. This is due to the fewer message flights needed to be exchanged in the session resumption, resulting in less computations at the sensor-side. The processing time for the certificate-based DTLS handshake[7] is considerably higher than both the symmetric key-based and the session resumption-based modes. The certificate-based DTLS requires about 5690 ms at the sensor-side which is mainly due to the expensive public key-based operations. Public key-related operations are the main contributor of sensor-side processing. In this work, there are three classes of public key-related computations. Elliptic Curve Diffie-Hellman (ECDH), the key agreement protocol. ECDH is a key agreement protocol which allows two parties, each having a publicprivate key pair, to establish a shared secret over an insecure channel. ECDH requires in average 437 ms and the deriving of a shared key point requires with 863.2 ms. Elliptic Curve Digital Signature Algorithm (ECDSA) is used for signing the server key exchange message and verifying the certificate message. The

Table 3: Client-side and sensor-side performance analysis of different DTLS modes to provide end-to-end security

	Sensor-side Processing Time (ms)	Sensor-side Energy Consumption (mJ)	Client-side Processing Time (ms)	Client-side Run-time (ms)
DTLS Session Resumption Without Server-side State ($DTLS\_Session\_Resumption\_WITH\_AES\_128$) (This Work)	160	8.87	45	205
Certificate-Based DTLS[7] ($DTLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_SHA\_256$)	5690	315.79	3744	9434
Symmetric key-Based DTLS[4] ($DTLS\_PSK\_WITH\_AES\_128\_CCM\_8$)	180	9.99	49	229

Table 4: Data handover latency between smart gateways with different packet size

Packet Size (byte)	Data Handover Latency (ms)
10	2.288
50	2.517
100	2.884
500	3.342
1K	3.685
5K	4.588

ECDSA signature requires in average 508.3 ms, whereas the ECDSA signature verification requires with in average 1896.5 ms. This shows how important it is to delegate such expensive operations through session resumption.

5.3.2. Sensor-side Energy Consumption

Similar to the previous section, energy consumption of each Wismote sensor node when performing end-to-end communication is computed using the aforementioned equation. We calculate the energy consumption of the Wismote sensor when performing the DTLS session resumption, the symmetric key-based DTLS, and the certificate-based DTLS. Results presented in Table 3 show that our techniques are considerably more energy efficient in comparison to the certificate-based DTLS[7] technique. It saves 11% of energy compared to the symmetric key-based DTLS[4].

5.3.3. Client-Side Processing Time

The total processing time at the client-side (end-user) using three different approaches is shown in Table 3. For the client-side, we used a machine with $IntelCore^{TM} i5 - 4570$ CPU operating at 2.2 GHz and having 6 GB of RAM. The processing time of our scheme using DTLS session resumption is 45 ms, where as the conventional symmetric key-based[4] requires 49 ms. This is due to the lesser number of control messages needed for session resumption, compared to the full symmetric key-based DTLS. The processing time for certificate-based DTLS handshake[7], is considerably higher than both the symmetric key-based and the session resumption-based modes. The certificate-based DTLS requires approximately 3744ms at the client-side which is mainly due to the expensive public key-based operations. Compared to symmetric key-based and certificate-based DTLS, our session resumption-based scheme has 8.1% and 98.7% improvements in terms of client-side processing time, respectively.

5.3.4. Client-Side Run-time Performance

Run-time refers to the time it takes for the handshake between the medical sensor node and the end-user to be done successfully. To provide end-to-end security, we calculate the total run-time of three different DTLS modes. As can be seen from Table 3, our scheme which exploits the DTLS session resumption technique is about 97% and 10% faster than certificate-based[7] and symmetric key-based DTLS handshake[4], respectively.

5.3.5. Data Handover Latency Between Two Smart Gateways

To demonstrate how our end-to-end security scheme enables mobility, we implemented a real system in which two UT-GATE gateways are employed. It is assumed that these gateways are connected through the fog layer where one of the gateways acts as a client and the other one acts as a server. In the experiments, we created a 100-byte lookup table for each gateway that consists of: i) Control data including the DTLS session resumption state, information about the authorized caregivers, medical sensors' IDs, and patients' IDs, ii) Patients' health data We computed the latency of the data handover process between the gateways. To show the scalability of our method, we considered messages with different sizes which may need to be exchanged between the gateways for the data handover process. As Table 4 presents, data handover latency between two gateways is negligible and mobility is supported in an agile way. In addition, by increasing the packet size, latency marginally increases showing the scalability of our scheme.

5.3.6. Memory Footprint

The memory footprint for symmetric key based DTLS, DTLS session resumption and certificate-based DTLS approaches are analyzed using *msp430-size* tool. For a more detailed information regarding the contribution of each components to static RAM and ROM the tool *msp430-objdump* is used. The results of our evaluation show that the certificate-based DTLS handshake is very expensive for resource-constrained sensor nodes. While, our DTLS session resumption approach requires similar resources as the symmetric key-based DTLS mode. Symmetric key-based DTLS requires 7.79 KB of RAM and 47.23 KB of ROM and our DTLS session resumption approach requires 8.25 KB of RAM and 47.86 KB of ROM. In DTLS session resumption approach, the RAM is just about 0.46 KB higher than symmetric key-based DTLS. This is due to a somewhat larger packet buffer size of DTLS session resumption approach. The certificate-based DTLS approach has the highest memory footprint With 12.32 KB of RAM, that is, 4.53 KB higher than symmetric key-based DTLS mode and 75.98 KB of ROM. This additional value is composed of more RAM requirements for larger packet buffers, session security parameters, certificate and buffering ECDSA signature values. Relic, requires 20.82 KB byte of ROM and and 1.49 KB of RAM. Relic cryptographic toolkit only appears in the certificate-based DTLS approach which makes it the major ROM and RAM contributor of this approach.

Table 5: Detailed Memory footprint of the three different DTLS approaches

	Symmetric Key-Based DTLS[4]		DTLS Session Resumption (This Work)		Certificate-Based DTLS[7]	
Modules	RAM (KB)	ROM (KB)	RAM (KB)	ROM (KB)	RAM (KB)	ROM (KB)
Relic Toolkit	-	-	-	-	1.49	20.82
AES-CCM	0	3.79	0	3.79	0	3.79
SHA2	0.29	2.48	0.29	2.48	0.29	2.48
DTLS-Client	0.22	0.27	0.22	0.27	0.6	0.27
DTLS-Server	0.008	0.21	0.171	0.21	0.42	0.21
Certificate Handler	-	-	-	-	0.02	1.46
DTLS	2.11	9.71	2.75	10.34	5.14	15.91

Symmetric cryptographic primitives of the three approaches that comprises of AES-CCM and SHA2 requires for 6.27 byte of ROM and 0.29 KB of RAM. The similarity is due to the fact that all the three approaches, employ the same symmetric primitives without further modifications. The portion labeled as DTLS in Table 5 is comprises of DTLS handler, state machine and re-transmission modules. As for the DTLS, symmetric key-based DTLS requires 9.71 KB of ROM and 2.11 KB of RAM, our session resumption approach requires 10.34 KB of ROM and 2.75 KB of RAM and 15.91 KB of ROM and 5.14 KB of RAM are required in the certificate-based DTLS, respectively. Certificate handler also appears only in the certificate-based DTLS approach which requires 1.46 KB byte of ROM and and 0.02 KB of RAM. Finally, the rest of RAM and ROM memories are dedicated to stack sizes and the Contiki OS.

6. Conclusions

We analyzed the performance of end-to-end security schemes in healthcare IoT systems. Based on the analysis, we distinguished that our scheme has the most extensive set of performance features in comparison to state-of-the-art end-to-end security schemes. Our end-to-end security scheme was designed by generating ECG-based cryptographic keys for medical sensor devices, certificate-based DTLS handshake between end-users and smart gateways as well as employing the session resumption technique for the communications between medical sensor devices and end-users. Our performance evaluation revealed that, the ECG signal based cryptographic key generation method that is employed in our end-to-end security scheme is on average 1.8 times faster than existing similar key generation approaches while being more energy-efficient. Compared to existing end-to-end security approaches, our scheme reduces the communication overhead by 26% and the communication latency between smart gateways and end users by 16%. Our scheme performed approximately 97% faster than certificate-based and 10% faster than symmetric key-based DTLS. In terms of memory requirements, certificate-based DTLS needs about 2.9 times more ROM and 2.2 times more RAM resources than our approach. In fact, the ROM and RAM requirements of our scheme are almost as low as in symmetric key-based DTLS. Our scheme is a very promising solution for ensuring secure end-to-end communications for healthcare IoT systems with low overhead. Our future work focuses on the trade-off analysis between security level and cost of the end-to-end security schemes in terms of latency and energy consumption.

References

1. S. R. Moosavi et al. End-to-End Security Scheme for Mobility Enabled Healthcare IoT. *Future Generation Computer Systems*, 2016.
2. E. Rescorla et al. Datagram Transport Layer Security (DTLS) Version 1.2. 2012.
3. R. Hummen et al. Delegation-based Authentication and Authorization for IP-based Internet of Things. In *11th IEEE International Conference on Sensing, Communication, and Networking*, pages 284–292, 2014.
4. Z. Shelby et al. CoRE Resource Directory. Internet-draft, 2017.
5. J. Granjal et al. End-to-end transport-layer security for Internet-integrated sensing applications with mutual and delegated ECC public-key authentication. In *International Conference on Networking*, pages 1–9, 2013.
6. N. Kang et al. ESSE: Efficient Secure Session Establishment for Internet-integrated Wireless Sensor Networks. *International Journal of Distributed Sensor Networks*, pages 1–11, 2016.
7. K. Hartke. Practical Issues with Datagram Transport Layer Security in Constrained Environments. Internet-draft, 2014.
8. A. M. Rahmani et al. Smart e-health gateway: Bringing intelligence to internet-of-things based ubiquitous healthcare systems. In *12th Annual IEEE Conference on Consumer Communications and Networking*, pages 826–834, Jan 2015.
9. C. Poon et al. A Novel Biometrics Method to Secure Wireless Body Area Sensor Networks for Telemedicine and m-Health. *IEEE Communications Magazine*, 44(4):73–81, 2006.
10. S. R. Moosavi et al. Cryptographic key generation using ECG signal. In *14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1024–1031, 2017.
11. S. R. Moosavi et al. Low-latency Approach for Secure ECG Feature Based Cryptographic Key Generation, year=2017. *IEEE Access*.
12. S. R. Moosavi et al. SEA: A Secure and Efficient Authentication and Authorization Architecture for IoT-Based Healthcare Using Smart Gateways. *Procedia Computer Science*, 52:452 – 459, 2015.
13. G. Zhang et al. Analysis of Using Interpulse Intervals to Generate 128-Bit Biometric Random Binary Sequences for Securing Wireless Body Sensor Networks. *IEEE Transactions on Information Technology in Biomedicine*, 16(1):176–182, 2012.
14. SmartRF06 Evaluation Board. http://www.ti.com/lit/ug/swru321a [accessed 2017-12-24].
15. Arago Systems. Wismote. http://www.aragosystems.com/en/document-center [accessed 2017-12-24].
16. O. Bergmann. TinyDTLS. http://sourceforge.net/p/tinydtls [accessed 2017-12-24].
17. D. Aranha et al. RELIC is an Efficient Library for Cryptography. http://code.google.com/p/relic-toolkit/ [accessed 2017-12-24].
18. A. Goldberger et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.