A Secure End-to-End IoT Solution

Avijit Mathur[1], Thomas Newe[†1,2], Walid Elgenaidi[1], Muzaffar Rao[1,2], Gerard Dooly[1,2] and Daniel Toal[2]

[1]Optical Fibre Sensors Research Centre,
[2]Mobile and Marine Robotics Research Centre,
Department of Electronics and Computer Engineering,
University of Limerick, Limerick, Ireland

[†]Corresponding author. Tel.: +353 61 202092. Fax.: +353 338176.
Email address: Thomas.Newe@ul.ie
[*]HKDF - HMAC based Key Derivation Function, [**]ECDH - Elliptic Curve Diffie-Hellman

Highlights

- An IoT system connecting sensors to any PC in the world is proposed.
- System deploys data security, routing security, key management and cloud security.
- Fast time and low energy consumption throughout the system.

Abstract: The Internet of Things (IoT) has been expanding in recent years with advancements in technologies, techniques and devices. This expansion has led to several different applications in the medical, civil, marine, military and domestic domains. Each of these domains have different requirements and challenges, with one common denominator: data security. Data security is an important aspect for any IoT network, however, in modern IoT systems simple data security may be not sufficient. This paper looks at a secure end-to-end IoT solution that allows wireless sensors/devices to connect to any PC in the world while guaranteeing data and network security. The scheme proposed in this paper can protect an IoT solution against several attacks like data breach, Denial of Service (DoS) and unauthorized access. Results obtained show that the technologies implemented, or used are superior in terms of time and energy consumption when compared to their counterparts in previously published works.

Keywords: *Secure IoT; Sensor Networks; Data Breach; HKDF[*]; ECDH[**]; WSN Security*

1. Introduction

There has been considerable advancement in the industry of IoT and Wireless sensor networks (WSN) for different purposes and applications. Each domain has different specifications, purpose, challenges and security requirements. For instance, a patient monitoring system [1,2] may require higher security than a smart parking solution. This is because a patient monitoring system handles patient data and sensor commands, which may be misused by compromising data, denying service and gaining unauthorized access to the network. Keeping this in mind, this paper focusses on securing a patient monitoring system for different components and aspects of the network, thereby providing the system with confidentiality, integrity and availability (CIA triad) [3].

Figure 1 illustrates the proposed system model for patient monitoring based on an end-to-end secure IoT. Here, several communication standards are used for the different devices. The standards are Message Queuing Telemetry Transport (MQTT) [4], universal asynchronous receiver/transmitter (UART) and RIME [5]. The devices are:

(1) Subscribed PC: a PC subscribed to a particular MQTT topic through the cloud network. This PC runs the Linux operating system Ubuntu 14.04 LTS and a custom python program to communicate with the cloud.

(2) Intel Edison Gateway [6]: This device runs the Debian (Ubilinux) OS and a python program to communicate with the cloud/sensors.
(3) Openmote [7]: This device runs the Contiki OS [8] and is used for the WSN side of the network.

The advantages of using a secure IoT system for patient monitoring systems are: (1) It allows internet connectivity for data to be send anywhere across the world. (2) The hospital staff / doctor can take immediate action if required. For example, sending an ambulance or administering a particular drug dose. (3) Patients have mobility as they can get their vitals checked while on-the-go, and (4) the confidence of the patients and the hospital staff in the use of these systems is increased because it is secure against data breach, denial of service and unauthorized access.

The rest of the paper is structured as follows: in Section 2, current solutions to the different aspects of IoT systems and key management are discussed. In Section 3, the overview of the IoT system, devices used, communication standards employed, and security technologies integrated are outlined. Additionally, the applicability of the Confidentiality, Integrity and Availability (CIA) triad to the proposed IoT system is discussed. In Section 4, several key management techniques like HMAC-based Key Derivation Function (HKDF) and Elliptic Curve Diffie-Hellman (ECDH) are defined and used in the proposed system. Results are provided for a real-world implementation of the IoT system shown in Fig. 1. In Section 5, the approaches for seeding a pseudo random number generator (PRNG) are discussed. Finally, the conclusion summarises the whole system with focus on security and the advantages it offers.

2. Related Work

Before proceeding into the current solutions related to our system presented in this paper, we will outline 'Our System' as mentioned in Table I. This refers to the system presented in Fig. 1 i.e. an end-to-end secure IoT system. The security technologies that are involved here are ECDH and HKDF algorithms, and AES Galois Counter Mode (AES-GCM 256) cipher. Each of these are described in detail in Section 4, where the corresponding experiments were carried out, and the results thus obtained from that section are combined and integrated into Table I under the 'Our System' heading. This is done for easier comparison with the other solutions provided in this section.

Now, moving the focus to the other current solutions available, there have been several different works that review the systems related to IoT/WSN fields. These systems may be harmed through different security attacks and there has been considerable amount of research into these systems and their defence mechanisms. For example the authors in [9] formulate a secure data transmission protocol, and discuss issues relating to integration of security in healthcare-based WSNs. In [10] the authors provide a lightweight and secure system for medical monitoring. The implementation makes use of limited resources hardware, and secures the data transmission through the system. Additionally, their implementation provides access control for authentication purposes. Other security related healthcare WSN applications can be found in [11–13].

Moreover, several works have tried to solve the security challenges associated with the IoT/WSN field. Reference [14], provides an overview on the challenges of IoT applications in smart grids related to security and privacy. In [15], the authors provide a key management architecture for IoT with resource constraints. They implement an IoT system with a client, a trusted authority, and a resource server to evaluate the key derived from their key management scheme. They have made use of the Contiki OS and the CC2538DK [16]. Their findings show that a complete handshake duration required to derive a key is 511.65ms to 711.11ms, compared to the key derivation schemes: HKDF and ECDH, used in our system which take 1.22ms and 682.6ms respectively. In addition, the energy consumed by the process in [15] ranges from 200 to 1000 μJ. In contrast, the key derivation processes in our system i.e. HKDF and ECDH schemes consume 49μJ and 21.49mJ respectively, see Table I.

In [17], the authors compare and contrast existing security schemes for wireless sensor networks (WSN), and introduce a new hybrid scheme. This scheme makes use of multiple key management concepts like trusted third party (TTP), node joining / revocation, and storage of keys. In contrast, the scheme presented in this paper does not require a TTP as authentication is provided through a hash-based mechanism discussed in our previous work

in [1]. Moreover, the scheme in [17] uses RSA to encrypt and decrypt a newly generated key. This can be expensive in the application layer, (up to 2 seconds). The usage of ECDH in our scheme to generate new keys takes only 682.6ms, Table I.

In [18], the authors introduced an optimised implementation of the NIST P-192 Elliptic Curve, running on an ATMega 128 Processor at 7.37MHz. The ECDH scheme implemented in [18] consumes 12.08 x 10^6 to 12.86 x 10^6 clock cycles, and 42mJ of energy. In contrast, the ECDH scheme used here, which is based on the ECC library of contiki [19], uses the NIST P-256 Elliptic Curve, running on the CC2538 SoC [20] with a clock frequency of 16MHz. This results in a consumption of 10.92 x 10^6 clock cycles, and 21.49mJ of energy as shown in Table I. Thereby showing an improvement of 1.16 to 1.94 x 10^6 clock cycles and 20.51mJ in energy consumption.

Reference [21] proposes new seeding techniques as a strong foundation for securing an IoT system. They provide in-depth detail on leveraging sensor data for providing secure seeds for a random number generator (RNG), which includes the Blum-Blum-Shub scheme, Washed + Rinsed scheme, Hashing algorithm and raw accelerometer data. In the proposed system, a real-time clock and accelerometer readings are used. These were timed and it was found that the latter is faster than the former by more than 80 times.

Finally, work related to HKDF, can be found in [22]. Here, the authors use the key derivation function for generating keys in an LTE network with the help of SHA-256 and HMAC. The paper lacks any timing or energy consumption measurements. In addition, the authors in [23] explain the security associated with HKDF and provide extensive comparisons with other key derivation functions.

3. System Model

3.1. Platform: Openmote

In the system presented here the openmote platform [24] is used. The main reason for using openmote is that it has sufficient RAM and ROM: 32kB and 512kB, and has hardware accelerators available for SHA-256, ECC and AES. This is useful as it speeds up the time required for computations for each of these security schemes, see Table II. In addition, the openmote platform supports multiple Operating Systems, including the Contiki OS [8], which is used in this work.

3.2. Platform: Intel Edison

In addition, the Intel Edison Arduino Breakout Board is utilised. The board is useful as it provides features like 1 GB RAM, and a USB Type A connector. The former is useful to facilitate simultaneous MQTT and UART connections, and the latter is useful for connecting the Openmote BaseStation directly to the Edison board. In addition, the Debian ubilinux is used instead of the default Yocto Linux. This is done to provide more freedom during programming, and for certain features like the paho MQTT client for python.

3.3. Platform: PC

A PC is used for receiving the encrypted data and decrypting it using AES-GCM 256, which is based on modifications to the library provided by the author in [25].

3.4. Platform: Cloud

Finally, the services of the eclipse paho project from [26] are also used. The MQTT running over this public server makes use of the 1883 unencrypted port. In the future it is planned to use the encrypted port 8883 in conjunction with transport layer security (TLS).

3.5. Overview:

The system model used defines an end-to-end system that is capable of connecting Internet of Things (IoT) enabled sensors on a patient to any PC in the world while avoiding direct access to the sensors from the internet. This technique is advantageous because it mitigates unauthorized IP-access of the sensors or routers from the internet, which may lead to various security attacks [27,28] while keeping the sensors, and routers connected to the internet.

Now, the actual sensors attached to the patient can be of any type like ECG, EEG, Pulse Oximeter, and so on. This is because the system model is structured such that it will work on any sensed data coming from the above-mentioned sensors attached to the sensor nodes (*openmotes*). This data is digital in nature as it has already been

converted by the analog-digital converter (ADC) of the sensor nodes (*openmotes*). For testing purposes, the system sensor nodes (*openmotes*) sense the temperature, humidity, and light (LUX) values with their on-board sensors. Additionally, when this data is being transmitted across the network each packet can carry a payload (data) of maximum size equal to 80 bytes. This is more than enough because each sensed data arising from the sensors may be in the order of a few bits.

As mentioned previously the architecture comprises of three communication standards i.e. MQTT, UART and RIME, see Fig. 1. MQTT provides a lightweight publish/subscribe model for communication between the Gateway and a Subscribed PC (PC subscribed to a MQTT Topic). It also has a small overhead for message transport, provides Quality of Service (QoS), and makes use of TCP/IP for network connectivity [4]. RIME, on the other hand, is a layered communication stack with low complexity of implementation for sensor network protocols, and low memory footprint (100-226 bytes) [5]. This is ideal for the wireless sensors' side of the system due to their resource constraints. Additionally, the RIME communication standard makes use of the chameleon architecture, which sits on top of the IEEE 802.15.4 MAC layer, which is related to the ZigBee specification as mentioned in [5].

Now, moving towards the work flow of the system as illustrated in Fig. 2, the system follows the following steps:

(1) The subscribed PC sends a 'start' command to the BS in the IoT/WSN network.
(2) The BS initiates secure on-demand routing protocol [1,13] to set up a route with Router R.
(3) Following this, ECDH phase I commences (Section 5).
(4) Router R broadcasts a 'Hello' packet with its public key (Q_r) to the sensors.
(5) Following the cluster head (CH) elections [1][29], the second phase of the ECDH (Section 5) commences. This results in the generation of a shared key (K_s) between the router and the sensors.
(6) Now, the sensors commence data sampling, Fig 3.

(7) Each sensor *i*, encrypts its sampled data D_i using AES-GCM 256bit

$$\{D_i\}K_s \,||\, T_s \qquad (1)$$

Where K_s is the session key, and T_s is the generated authentication tag.

(8) Each sensor's encrypted data is aggregated at the CH, and send to Router R. The aggregation takes place in the following manner:
$\{D_1\}K_s \,||\, T_s \,|\, \{D_2\}K_s \,||\, T_s \,|\, \ldots \,|\, \{D_n\}K_s \,||\, T_s$ **represented as** $[\{D_i\}K_s \,||\, T_s]_n$ **in Fig 3.** **(2)**
Here, n is the number of sensors in the cluster
(9) Following this, Router R decrypts the data and verifies its authentication tag.
(10) R then encrypts this aggregated data with a pre-loaded key K_m, and sends it to the BS.

$$\{D_1 \,||\, D_2 \,|\, \ldots. \,|\, D_n\}K_m \,||\, T_m \qquad (3)$$

(11) The BS, forwards this to the PC via the gateway.
(12) Finally, the PC verifies tag T_m, and decrypts the encrypted data using AES-GCM with the pre-loaded key K_m.

As shown, data / commands travel from different entities (sensors, routers, Gateway, Cloud, and PC), and through different communication channels (RIME, Internet, and UART). This raises security concerns that call for extensive measures in order to safeguard the system, the data, and the people involved. The different security measures implemented in our system are: (see Fig. 1.)

(1) Confidentiality: AES-GCM 256 bit [30].
(2) Data Integrity: Using AES-GCM's one-pass authentication.
(3) Entity Authentication between the BS and the Router R using our HASH-based scheme [1]. Authentication between PC and the Gateway G using pre-loaded keys.

(4) Key Management using ECDH [31] and HKDF [32] between the Router R and the sensors, and between the PC and Router R, respectively.

These measures, and their usage in our system are described in the following section (Section 3.6).

3.6. Security Overview (CIA Triad)

Confidentiality refers to the protection of data from unauthorized users or malicious individuals, the lack of which can have serious repercussions on the smooth working of a system. In the system presented here, confidentiality is provided by making use of the AES - Galois/Counter Mode (GCM) [30] with 256-bit security. This mode has several advantages over other cipher modes [33], such as:

(1) Accepts arbitrary length initial-vector (IV).
(2) Provides one-pass data authentication.
(3) Can be used as a stand-alone MAC or incremental MAC.

The authors in [33] show how the GCM mode is secure after using the above mentioned features. Thereby, increasing our confidence in the usage of this mode for the system presented.

Integrity refers to the protection/detection of data from any modification. To provide this, the system makes use of the previously mentioned, AES-GCM 256. This approach is useful as it allows for one-pass data authentication with confidentiality.

Authentication refers to the authorisation/verification of an entity that wishes to join the network. It also allowing a network device to verify the authenticity of a data/packet sender. This is achieved in two parts:

(1) BS to router R: a SHA-256 hash-based scheme is implemented. This scheme helps the router to verify the authenticity of the BS [1].
(2) PC to the BS via Gateway: Currently, a preloaded-key K_m is used with a randomly generated salt *S*. However, in the future it is planned to use the transport layer security (TLS) authentication protocol as this can help prevent tampering, message forgery and eavesdropping [34]. TLS has yet to be implemented in the proposed system.

4. Key management

All the above processes are fruitless if the keys are not updated securely. This requires the use of key management, which is important for the wireless network in the implemented system as it may be exposed to different types of security attacks. So, the key management implemented here allows for the keys between two parties to be updated without disclosure of any associated private data. This update ensures that any compromised key may not be used to attack the system.

The process of key management entails the following parameters: key generation, key updates, and key revocation. This section focuses primarily on the key update process between: (1) Router *R* and the Cluster of sensors *S*, and (2) Subscribed PC *P* and Router *R*.

Following notations are used in this section:

a) K_s: Session key generated between *R* and *S*
b) K_m: Pre-loaded master key between the *P* and *R*
c) K_{m_new}: New Key generated between *P* and *R*

K_s key is used to encrypt and authenticate data between the router *R* and cluster of Sensors *S* using the AES-GCM 256 cipher. This key is always generated using the ECDH algorithm. K_m key between the *P* and router *R* is pre-loaded for first time use. Following this, it is always updated using the HKDF mechanism, which results in a new session key K_{m_new}

4.1. Key Updates

The key management process is implemented in the system in two phases:

(1) Router *R* to the Cluster of sensors *S*: Here, the key management process encompasses key generation, and key update, with usage of the ECDH algorithm [31]. When compared to its counterparts like RSA [35], Elliptic curve cryptography (ECC) is used to ensure that the key size is small [36], and implementation is resource efficient [37]. Elliptic curve NIST P-256 curve (secp256r1) is selected as the curve, as it is one of the NIST recommended curves [38] in the 256 bit ECC family. This curve is generated from random values, and is implemented in the ECC library provided in Contiki [19].

Following is a brief explanation of the implemented mechanism:

Phase 1

Phase 2

a) Router *R* calculates shared secret (K_s) = $k_r.Q_c$ (*Fig. 4, point 5. of Router*):

$$\textbf{Here Since } Q_c = k_c.G\textbf{, therefore } K_s \textbf{ can also be written as } k_r.k_c.G \quad (4)$$

b) Sensors *S* calculate shared secret (K_s) = $k_c.Q_r$ (*Fig. 4, point 5. of Cluster of Sensors*):

$$\textbf{Here Since } Q_r = k_r.G\textbf{, therefore } K_s \textbf{ can also be written as } k_c.k_r.G \quad (5)$$

Now, Eq. 4 and Eq. 5 show that the shared secret (K_s) is same for both parties. This is because the scalar multiplication (.) observes the commutative law of mathematics. Thereby making $\mathbf{k_r.k_c.G}$ **and** $\mathbf{k_c.k_r.G}$ equal. Therefore, it is clear that the new session key generated K_s is same on both sides (from Fig. 4, Eq. 4, and Eq. 5). This session key is used to encrypt data from the sensors to the router using AES-GCM 256 bit, and is periodically updated. This update time is set arbitrarily and can be adjusted to suit the needs of the application.

(2) Router *R* to PC side: This part of the network implements the HKDF process in accordance with the IETF standard RFC 5869 [32], and the HMAC process in accordance with the IETF standard RFC 2104 [39]. In these implementations, the SHA hardware accelerator for cc2538 in the openmote is used, and python programming language in the PC is used.

Now, the reason for using HKDF instead of ECDH is that the latter would have to be implemented across different platforms running different operating systems, and the exchange would have to take place through the cloud using MQTT (see Fig 1). This would put an excessive workload on the IoT router *R* that is already handling the ECDH process with the cluster of sensors *S*.

Moving on, the designated router *R* updates its key using HKDF, when it receives an '*update*' command from the subscribed PC via. the Gateway (*using MQTT protocol*) as shown in Fig. 5. The process results in a new key K_{m_new} generated on both sides (PC and *R*) without sending any information over the network. This saves energy because radio transmissions are reduced as compared to ECDH, see Table III. However, the disadvantage is that if the router or the PC is compromised, then the subsequent keys generated may be compromised. So, in-order to reduce this likelihood one may change the 'opad', and 'ipad' values used in HMAC to secret values instead of the general ones provided in the HMAC RFC 2104 [39]. In this fashion, an attacker may not be able to launch attacks that analyse the cipher text, unless they have knowledge of both the 'opad' and the 'ipad'.

4.2. Results

This section presents the results associated with some of the key management techniques mentioned previously, and other cryptographic measures used in our system. These are compared with each other and/or with other systems as mentioned in their respective tables.

Table III presents analysis of different key management techniques in our presented system, in TinyECC [40] implementation, and in the system presented in [18]. From the table, it can be deduced that the non-optimized ECDH implemented in contiki is more efficient than the one presented in TinyECC without optimizations enabled [40], and also to the one presented in [18].

Additionally, from Table III it is seen that HKDF outperforms ECDH in terms of time and energy. This is because the HKDF makes use of only HMAC as compared to ECDH that makes use of expensive operations like scalar multiplication and radio communications. However, when considering security, ECDH outperforms HKDF. This is because the HKDF makes use of pre-loaded session key, as compared to ECDH, which generates the first session key through a Diffie-Hellman Exchange.

Table IV presents the measurements for the AES-GCM 256 library in Contiki OS. The results show that this is approximately 80 times faster when compared to the AES-CBC 128 and AES-CBC 256 libraries in Waspmotes [17,41]. This shows the superior technologies and techniques used in the presented system provide security with fast time and low current consumption.

5. Pseudo Random number generator Seeding (PRNGS)

The previously mentioned key management schemes require a strong foundation. This may be achieved by strengthening the core random number generation process as it is used for initial random number generation at the BS [1] and for private key generation in ECDH. Therefore, in this system several techniques are discussed that can be used for the improvement of these processes.

Notations used in this section:

a) **RTIMER_NOW**: Current time in system ticks.
b) **srand**: 'C' library function from <stdlib.h>. This is used to initialise the PRNG with a seed.
c) **RAND_MAX**: Maximum value the random function (rand()) can return, and is equal to 2147483647.

Table V examines four scenarios with different initial seeding, and rand() function range. From the table, it is deduced that scenarios 3 and 4 are disregarded due to too much time consumption (*261 ms*), and low variation of power-values respectively. However, Scenario 3 is still included in the following test for further comparison because it results in changes to output random numbers, visible to the naked eye, unlike Scenario 4.

The following test measures Scenarios 1, 2, and 3 using the NIST's statistical test suite for random and pseudo random number generators [42]. The number of input bit stream length n is 10000, and the significance level α is set to 0.01, which means that 1 in every 1000 scenario may fail. Therefore, any proportion-value (P-value) ≥ 0.01 means the PRNG has passed that statistical test. Table VI shows the results with the P-value shown in terms of percentage. From the table, it is inferred that Scenario 2 performs better than the other scenarios even though it uses a PRNG. This is because the accelerometer in Scenario 3 has limited range and resolution, and the power consumption in Scenario 4 does not vary considerably, as mentioned previously. Moreover, the time consumed by Scenarios 1 and 2 is considerably less when compared to Scenario 3 ($\approx 1/500$).

Additionally, Scenario 2 performs better compared to the Raw Data and Secure Hash algorithm approach used in [21]. However, it falls behind when compared to the Washed+Rinsed technique and Blum-Blum-Shub scheme [21] as these schemes are oriented towards true random number generation.

6. Conclusion

Security and key management techniques related to the IoT have not been sufficient in current systems. This calls for the requirement to secure every component of an IoT system, otherwise it could lead to information leakage from the system. This will also affect the confidence of people involved with using these systems. Therefore, the presented IoT system in this paper tackles the problem of security from different angles.

First, confidentiality and data-authentication are added using the AES-GCM 256-bit cipher, which outperforms other schemes as seen in Table IV. Second, key management schemes are included with periodic key updates, command-based key updates and key-generation features. These schemes are supported by ECDH, HMAC and HKDF standards that outperform other schemes mentioned in this paper [18]. Third, the PRNG from Contiki's 'C' library is analysed using the NIST's statistical test suite [42]. This provided approaches for seeding the PRNG to improve its randomness, thereby, giving more insight into the importance of random number generators in key-management.

To conclude, the system presented in this paper provides a high level of security with cloud connectivity for IoT enabled devices. In addition, a working implementation on different technologies i.e. openmote, Intel Edison, Cloud and PC was presented.

Future Work

In the future, it is planned to incorporate TLS/SSL security on the cloud side of the system as this will add additional authentication security for the PC and the Gateway. This will ensure that the Gateway recognises the identity of the PC sending commands to the IoT network. This will assist in the prevention of address-spoofing and replay attacks.

Acknowledgments

This work was supported by the Irish Research Council (grant number IRC-GOIPG/2013/1132), the Strengthening Training and Research through Networking and Globalization of Teaching in Engineering Studies (STRoNGTiES) program and the SFI Centre for Marine and Renewable Energy Ireland (MaREI) (grant numbers 12/RC/2302 and 14/SP/2740).

References

[1] A. Mathur, T. Newe, M. Rao, Defence against black hole and selective forwarding attacks for medical WSNs in the IoT, Sensors (Switzerland). 16 (2016). doi:10.3390/s16010118.

[2] S. Möller, T. Newe, S. Lochmann, Prototype of a secure wireless patient monitoring system for the medical community, Sensors Actuators, A Phys. 173 (2012) 55–65. doi:10.1016/j.sna.2011.10.016.

[3] NIST, FIPS PUB 199: Standards for Security Categorization of Federal Information and Information Systems, Fips. 199 (2004) 13.

[4] International Business Machines Corporation (IBM), Eurotech, MQTT V3.1 Protocol Specification, (n.d.). http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html (accessed October 24, 2016).

[5] A. Dunkels, Rime-a lightweight layered communication stack for sensor networks., Eprints.sics.se. (2007). http://eprints.sics.se/963%5Cnpapers2://publication/uuid/76662E66-E6A9-4373-B64D-3576B45BE9D0.

[6] Intel Corporation, Intel® Edison Kit for Arduino, (2015). http://download.intel.com/support/edison/sb/edisonarduino_hg_331191007.pdf.

[7] U. Berkeley, openmote: open hardware for the internet of things, (2012). http://www.openmote.com/home.html (accessed March 27, 2015).

[8] A. Dunkels, B. Grönvall, T. Voigt, Contiki - A lightweight and flexible operating system for tiny networked sensors, in: Proc. - Conf. Local Comput. Networks, LCN, 2004: pp. 455–462. doi:10.1109/LCN.2004.38.

[9] S. Ben Othman, A.A. Bahattab, A. Trad, H. Youssef, Secure Data Transmission Protocol for Medical Wireless Sensor Networks, in: 2014 IEEE 28th Int. Conf. Adv. Inf. Netw. Appl., IEEE, 2014: pp. 649–656. doi:10.1109/AINA.2014.80.

[10] Daojing He, S. Chan, Shaohua Tang, A Novel and Lightweight System to Secure Wireless Medical Sensor Networks, IEEE J. Biomed. Heal. Informatics. 18 (2014) 316–326. doi:10.1109/JBHI.2013.2268897.

[11] H. Al-hamadi, A. Gawanmeh, M. Al-qutayri, Simulation Framework for a Security Protocol for Wireless Body Sensor Networks, in: 2016 IEEE 41st Conf. Local Comput. Networks Work., IEEE, Dubai, 2016: pp. 248–253. doi:10.1109/LCN.2016.055.

[12] S. Saha, S. Kumar Tomar, Issues in Transmitting Physical Health Information in m-Healthcare, Int. J. Curr. Eng. Technol. 3 (2013) 411–413.

[13] A. Mathur, T. Newe, Medical WSN: Power, routing and selective forwarding defense, in: T. Plank (Ed.), 13th Int. Conf. Telecommun., IEEE, Graz, 2015: pp. 1–6. doi:10.1109/ConTEL.2015.7231208.

[14] F. Dalipi, S.Y. Yayilgan, Security and Privacy Considerations for IoT Application on Smart Grids: Survey and Research Challenges, 2016 IEEE 4th Int. Conf. Futur. Internet Things Cloud Work. (2016) 63–68. doi:10.1109/W-FiCloud.2016.28.

[15] S. Raza, L. Seitz, D. Sitenkov, G. Selander, S3K: Scalable Security with Symmetric Keys - DTLS Key Establishment for the Internet of Things, IEEE Trans. Autom. Sci. Eng. 13 (2016) 1270–1280. doi:10.1109/TASE.2015.2511301.

[16] Texas Instruments, CC2538 Development Kit Quick Start Guide, (2013) 1–7. http://www.ti.com/lit/ml/swru347/swru347.pdf.

[17] W. Elgenaidi, T. Newe, E. O'Connell, G. Dooly, Trust security mechanism for maritime wireless sensor networks, Concurr. Comput. Pract. Exp. 22 (2016) 685–701. doi:10.1002/cpe.3945.

[18] Z. Liu, H. Seo, J. Groszschaedl, H. Kim, Efficient Implementation of NIST-Compliant Elliptic Curve Cryptography for Sensor Nodes Using 8-bit AVR Microcontroller, IEEE Trans. Inf. Forensics Secur. 11 (2014) 1–13. doi:10.1109/TIFS.2015.2491261.

[19] A. Dröscher, Texas Instruments, cc2538 ECDH Test Project, (2014).

[20] Texas Instruments, CC2538 Powerful System-On-Chip for 2.4-GHz IEEE 802.15.4, 6LoWPAN and ZigBee® Applications, Texas Instruments. (2012). http://www.ti.com/lit/ds/swrs096b/swrs096b.pdf (accessed February 7, 2015).

[21] S.L. Hong, C. Liu, Sensor-based random number generator seeding, IEEE Access. 3 (2015) 562–568. doi:10.1109/ACCESS.2015.2432140.

[22] F. Chen, J. Yuan, Enhanced key derivation function of HMAC-SHA-256 algorithmin LTE network, Proc. - 2012 4th Int. Conf. Multimed. Secur. MINES 2012. 3 (2012) 15–18. doi:10.1109/MINES.2012.106.

[23] H. Krawczyk, Cryptographic extraction and key derivation: The HKDF scheme, Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 6223 LNCS (2010) 631–648. doi:10.1007/978-3-642-14623-7_34.

[24] X. Vilajosana, P. Tuset, T. Watteyne, K. Pister, Openmote: Open-source prototyping platform for the

industrial IoT, in: Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng. LNICST, 2015: pp. 211–222. doi:10.1007/978-3-319-25067-0_17.
[25] B. Zhu, AES-GCM-Python, (2013). http://about.bozhu.me.
[26] The Eclipse Foundation, eclipse paho, Eclipse Found. (2016).
[27] Q. Gu, P. Liu, Denial of Service Attacks, in: H. Bidgolli (Ed.), Handb. Comput. Networks Distrib. Networks, Netw. Planning, Control. Manag. New Trends Appl. Vol. 3, John Wiley & Sons, Hoboken, NJ, 2007. doi:10.1002/9781118256107.ch29.
[28] A.D. Wood, J.A. Stankovic, Denial of service in sensor networks, Computer (Long. Beach. Calif). 35 (2002). doi:10.1109/MC.2002.1039518.
[29] A. Mathur, T. Newe, M. Rao, Healthcare WSN: Cluster Elections and Selective Forwarding Defense, Proc. - NGMAST 2015 9th Int. Conf. Next Gener. Mob. Appl. Serv. Technol. (2016) 341–346. doi:10.1109/NGMAST.2015.14.
[30] D. McGrew, J. Viega, The GCM Mode, 2004. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf%5Cnhttp://siswg.net/docs/gcm_spec.pdf.
[31] Certicom Research, Standards for Efficient Cryptography, SEC1: Elliptic Curve Cryptography, 1 (2009) 1–144.
[32] H. Krawczyk, IBM Research, P. Eronen, Nokia, HMAC-based Extract-and-Expand Key Derivation Function (HKDF), 2010.
[33] D.A. McGrew, J. Viega, The security and performance of the Galois/counter mode (GCM) of operation, Prog. Cryptol. - INDOCRYPT 2004. 5th Int. Conf. Cryptol. India. Proc. (Lecture Notes Comput. Sci. Vol.3348). (2004) 343–55. doi:10.1007/978-3-540-30556-9_27.
[34] S. Turner, Transport layer security, IEEE Internet Comput. 18 (2014) 60–63. doi:10.1109/MIC.2014.126.
[35] J.W. Bos, J.A. Halderman, N. Heninger, J. Moore, M. Naehrig, E. Wustrow, Elliptic curve cryptography in practice, IACR Cryptol. …. (2014) 157–175. doi:10.1007/978-3-662-45472-5_11.
[36] A.K. Lenstra, E.R. Verheul, Selecting cryptographic key sizes, J. Cryptol. 14 (2001) 255–293. doi:10.1007/s00145-001-0009-4.
[37] D.J. Bernstein, T. Lange, Security dangers of the NIST curves, (2013). http://www.hyperelliptic.org/tanja/vortraege/20130531.pdf.
[38] NIST, Recommended Elliptic Curves for Federal Government Use, 1999.
[39] H. Krawczyk, M. Bellare, R. Canetti, HMAC: Keyed-Hashing for Message Authentication, 1997. https://tools.ietf.org/html/rfc2104.
[40] A. Liu, P. Ning, TinyECC : A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks ∗, in: Proc. 7th Int. Conf. Inf. Process. Sens. Netw., 2008: pp. 245–256.
[41] Libelium Comunicaciones Distribuidas S.L, Waspmote Encryption Libraries Programming guide, (2015) 12–21. http://www.libelium.com/downloads/documentation/encryption_libraries_guide_eng.pdf.
[42] A. Rukhin, J. Soto, J. Nechvatal, S. Miles, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, A statistical test suite for random and pseudorandom number generators for cryptographic applications, Natl. Inst. Stand. Technol. 800 (2010) 131. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA393366.
[43] Keysight Technologies, 66321D Mobile Comm DC Source w/ Battery Emulation, DVM, (n.d.). http://www.keysight.com/en/pd-1000000821%3Aepsg%3Apro-pn-66321D/mobile-comm-dc-source-w-battery-emulation-dvm?cc=IE&lc=eng (accessed March 25, 2015).
[44] A. Dunkels, J. Eriksson, N. Finne, N. Tsiftes, Powertrace : Network-level Power Profiling for Low-power Wireless Networks Low-power Wireless, in: SICS Tech. Rep. T201105, 2011.

Authors's Biographics

Mr. Avijit Mathur (M.Eng, B.Tech) received his M.Eng degree in Information and Network Security from University of Limerick, Ireland (2013), and a B.Tech degree in Computer Science and Engineering from Amity University, India (2012). He has experience with computer programming and networking. Currently, he is pursuing his research on medical based wireless sensor network (WSN) systems. The research focusses on security, routing, energy-efficiency, and overall development of the WSNs for medical applications. His research interests include wireless networking, sensor networks, data link layer, cryptography, and routing attacks.

Dr. Thomas Newe is a Senior Lecturer in the Department of Electronic & Computer Engineering at the University of Limerick and group leader of the Wireless Sensor Network Security group. He has graduated nine PhD students in the general area of Security for Wireless Sensor Networks (WSNs) and currently has six registered students working in the areas of secure WSN for observation and monitoring in marine systems and medical environments. His areas of research include: IoT, Wireless Sensor Networks, Secure Marine Communications, Operating Systems, Cryptography/Encryption Algorithms, Data Security, Network Security and Formal Verification Methods.

Mr. Walid Elgenaidi is a Doctoral Researcher in Computer Engineering in the Department of Electronic & Computer Engineering at The University of Limerick. He holds a B.Eng. in Telecommunication Engineering from Banghazi University (2005), a Masters in Communication Engineering from University Technology Malaysia (2007). He is a currently member in The Optical Fibre Sensors Research Centre at University of Limerick. His research interest includes: Wireless Sensor Networks, Network Security and Security Protocol Design, Cryptography/Encryption algorithms, Programming languages and Image Processing System.

Dr. Muzaffar Rao obtained his B.E. degree in Electronics from Sir Syed University of Engineering and Technology, Karachi, Pakistan and M.E. degree in Electrical from NED University of Engineering and Technology Karachi, Pakistan in 2005 and 2009 respectively. Recently he received his PhD from The University of Limerick, Ireland. His research interests include Information Security and Cryptography, FPGA based System Designs, Hardware solutions of Cryptographic Applications and Digital Systems Design.

Dr. Daniel Toal is a chartered engineer in Electrical and Systems Engineering: (Hons Dip Elec Eng, Dublin Institute of Technology; BSc (eng) University of Dublin (TCD) 1984; MSc - Manufacturing Systems Engineering, Cranfield University, UK, 1986; PhD Marine Robotics, University of Limerick (UL) 2004). Daniel is currently an Associate Professor at UL and has taught: Automation, Robotics, Instrumentation, Avionics, Sensors, Electrical Machines. Daniel is the founder and director of the Mobile & Marine Robotics Research Centre at the University of Limerick and is Co PI of the SFI Centre MaREI - Marine Renewable Energy Ireland www.marei.ie.

Dr. Gerard Dooly is BEng Electronic and Computer Engineering Department, University of Limerick, 1999-2003. He has a PhD from Optical Fibre Sensors Research Centre, University of Limerick, 2003 – 2008 titled, “An Optical Fibre Sensor for the Measurement of Hazardous Emissions from Land Transport Vehicles. His research interests include optical fibre sensors, differential optical absorption spectroscopy, advanced control systems, underwater robotic engineering and advanced sonar operations and processing.

Figures

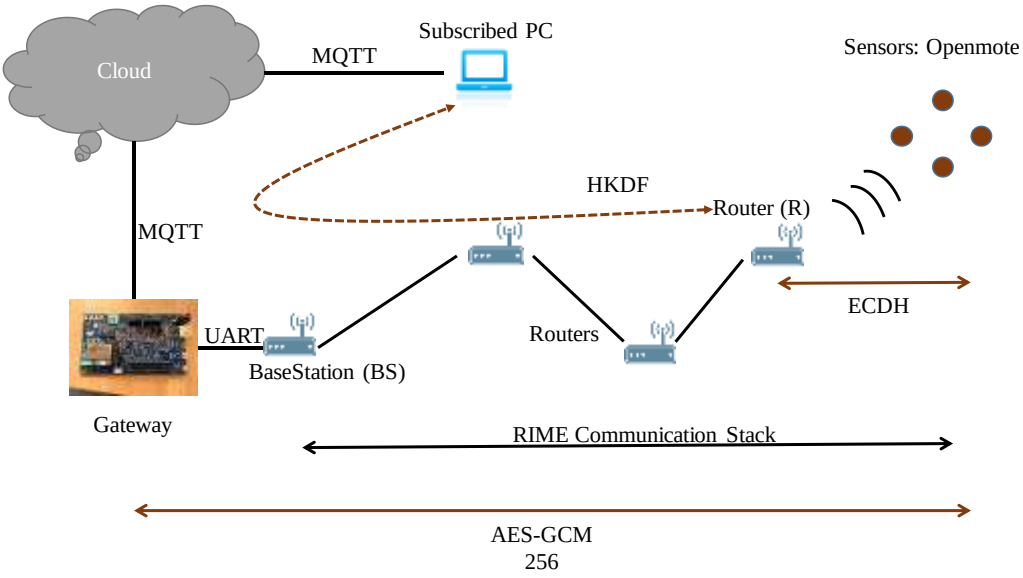

Fig. 1 End-to-End Secure IoT system: Communications standards used and security implemented

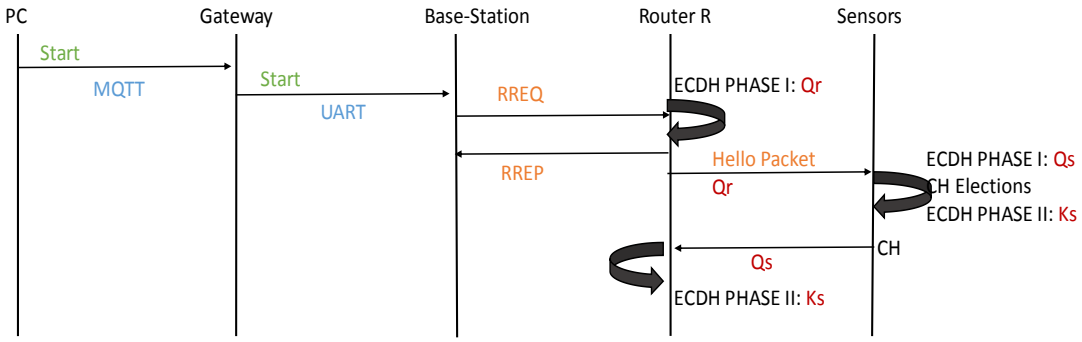

Fig. 2. Work Flow Diagram for the system - I

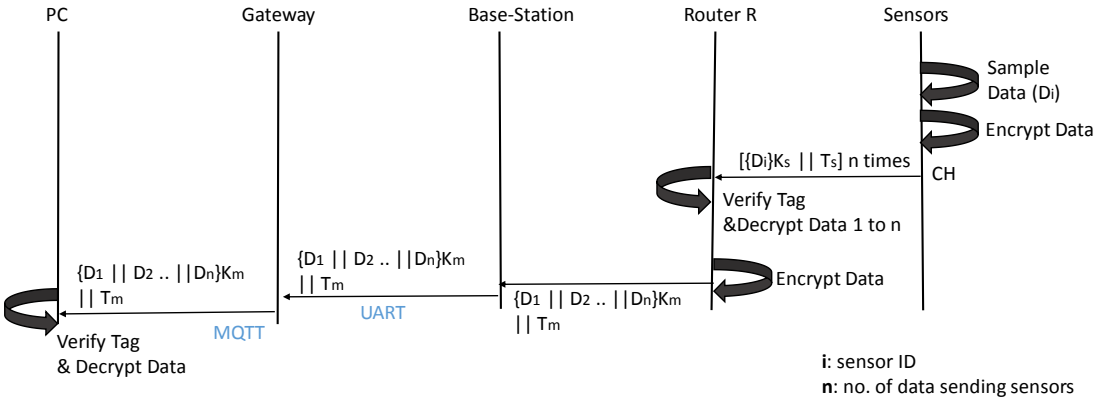

Fig. 3. Work Flow Diagram for the system - II

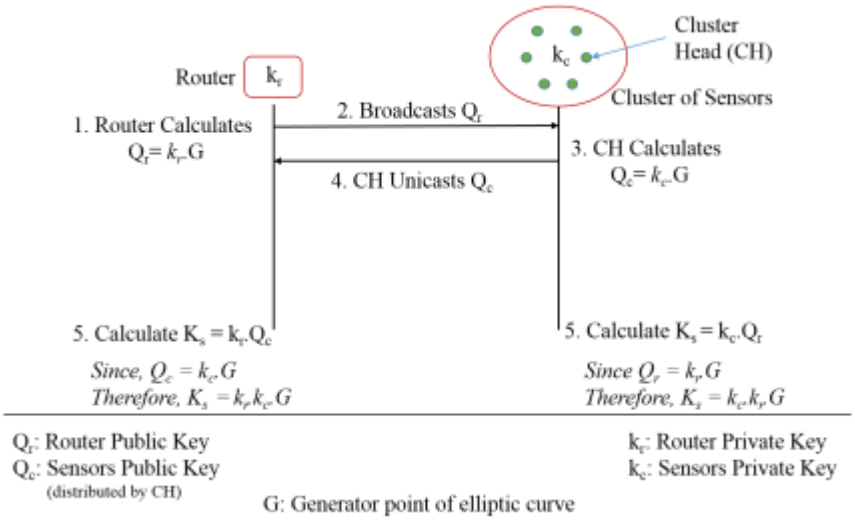

Fig. 4. ECDH between router and sensors.

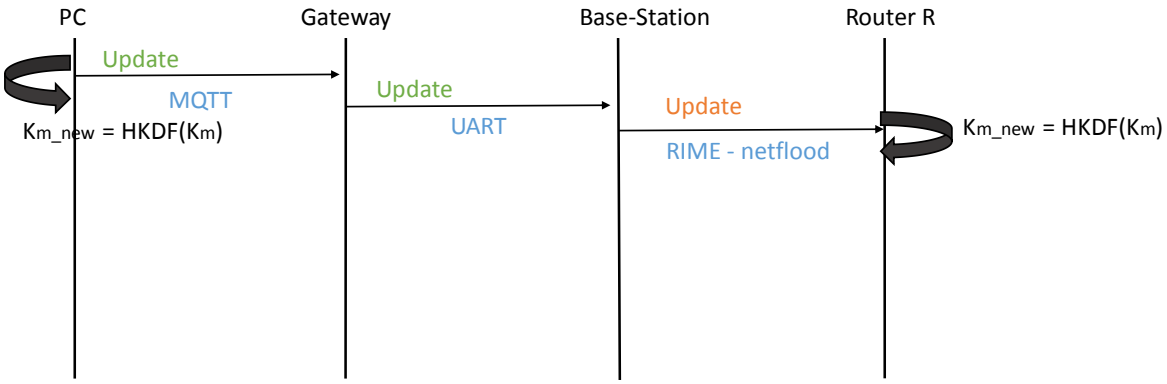

Fig. 5. Key Update between the subscribed PC and the Router R using HKDF.

Table I. Related Work Comparison

Related Work Ref.	**Scheme Used**	**Frequency (MHz)**	**Time (ms)**	**Energy (mJ)**
[18]	ECDH	7.37	--	42
[40]	ECDH	13	2718.35	80.05
Our System	ECDH	16	682.6	21.49
[17,41]	RSA-1024*	14.7456	26.494	1.8[1]
	AES-128-CBC*		10.44	0.6
Our System	AES-256-GCM	16	0.132	2.25[2]
[15]	S3K (DTLS based)	32	511.65 **to** 711.11	0.2 to 1
Our System	HKDF	16	1.2294	0.049

[1] Calculated using E = V * I * t equation, where V = 3V, and I was retrieved from datasheet.

[2] Calculated using E = V * I * t equation, where V = 3.28V and I was measured using Agilent 66321D device [43].

* Datasheet values

Table II. ECDH Phase I between Router R and Cluster of Sensors S

Router *R*	**Cluster of Sensors *S***
Choose Random Secret => Generate private key k_r	Private Key k_c pre-loaded
Calculate public key $Q_r = k_r.G^*$	Calculate public key $Q_c = k_c.G^{**}$
Broadcast Q_r to the cluster of sensors *S'*	CH of these sensors sends (unicasts) Q_c to the Router *R''*

G is the generator point of elliptic curve secp256r1

**Fig .4, Point 1, **Fig. 4, Point 3*
'Fig. 4, Point 2, ''Fig. 4, Point 4

Table III: Key Management Techniques Analysis

	ECDH **(Openmote 16 MHz)[1]**	***ECDH in*** [40] **(imote2 13MHz)[2]**	***ECDH in*** [18] **(Atmel 128) @ 7.37 MHz) [3]**	***HKDF[4]***	***HMAC[4]***
Libraries	ECC (built-in Contiki)	TinyECC	--	HMAC (custom-built)	SHA 256 (built-in Contiki)
Total Time (ms)	682.6	2718.35	1639 - 1743	1.2294	0.328
Clock Cycles (*10^6)	10.912	35.334	12.08 – 12.85	0.019	0.005
Time for each stage (ms)	N/A	Initialization: 0.04	--	--	Set Key: 0.0226
	Round 1: 343 Round 2: 339.6	Key-Establish: 2718.31	--	Extract: 0.599 Expand: 0.630	Initialization: 0.0466 Process: 0.281
Energy	21.49mJ	80.05mJ	42mJ	0.049mJ	0.013mJ
Notes	ECC hardware accelerator	--	--	Software + SHA hardware accelerator	Software + SHA hardware accelerator

[1]No optimizations, [2]Optimizations Disabled, [3]NIST P-192 Curve, [4]Custom build

Table IV: AES Analysis

Platform → ***Metrics ↓***	***Openmote AES – GCM – 256 @16MHz*** **(5 bytes' data)**	***Waspmote AES–128 CBC Zero padding @ 14.7456MHz*** [17] [41]	***Waspmote AES-256 CBC Zero padding @ 14.7456MHz*** [41]
Libraries	AES (built-in Contiki)	AES (built-in Wasp mote)	
Total Time (ms)	0.132	10.44	10.76
Time for Decrypt stage (ms)	0.0752	3.21	3.89
Time for Encrypt stage (ms)	0.0569	7.23	6.87
Current (mA)	5.2	7.4 to 64.9	22.4
Notes	AES hardware accelerator	Hardware + Software	Hardware + Software

Table V. PRNGS Scenarios

	Initial seeding	**rand() range**	**Remark**
Scenario 1	No seeding	0 to RAND_MAX	Results stored as characters, where every character corresponds to 1 byte
Scenario 2	Using srand: srand(*rand() % 256*)	0 to RTIMER_NOW	Results stored as characters, where every character corresponds to 1 byte.
Scenario 3	Values (μ) from Accelerometer Sensor	0 to μ	Too much time consumption (261 ms) *
Scenario 4	Power Values (β) obtained from Contiki's powertrace [44]	0 to β	Power values do not vary considerably for the scenario to work

*Time taken is large as the sensor is calibrated using **'adxl346.configure(ADXL346_CALIB_OFFSET, 0)'** function

Table VI. NIST Statistical Test Suite Results

*(*P-value shown in percentage)

Tests	**Scenario 1**	**Scenario 2**	**Scenario 3**
Frequency	21.33%	53.4%	0%
Block Frequency	91.14%	53.4%	0%
Cumulative Sum	73.99%	73.99%	0%
Runs	73.99%	91.1%	0%
Longest Run	73.99%	99.14%	0%
Rank	73.99%	21.3%	0%
FFT	6.68%	53.4%	0.04%
Linear Complexity	6.68%	53.4%	53.4%
Time Consumed	0.42ms	0.42ms	261ms