A Novel and Lightweight System to Secure Wireless Medical Sensor Networks

Daojing He, Sammy Chan, *Member, IEEE*, and Shaohua Tang, *Member, IEEE*

***Abstract*—Wireless medical sensor networks (MSNs) are a key enabling technology in e-healthcare that allows the data of a patient's vital body parameters to be collected by the wearable or implantable biosensors. However, the security and privacy protection of the collected data is a major unsolved issue, with challenges coming from the stringent resource constraints of MSN devices, and the high demand for both security/privacy and practicality. In this paper, we propose a lightweight and secure system for MSNs. The system employs hash-chain based key updating mechanism and proxy-protected signature technique to achieve efficient secure transmission and fine-grained data access control. Furthermore, we extend the system to provide backward secrecy and privacy preservation. Our system only requires symmetric-key encryption/decryption and hash operations and is thus suitable for the low-power sensor nodes. This paper also reports the experimental results of the proposed system in a network of resource-limited motes and laptop PCs, which show its efficiency in practice. To the best of our knowledge, this is the first secure data transmission and access control system for MSNs until now.**

***Index Terms*—Access control, data transmission, medical sensor networks, security.**

I. Introduction

Recently, with the rapid development in the wearable biosensor and wireless communication technologies, wireless medical sensor networks (MSNs) have emerged as a promising technique which will revolutionize the way of seeking healthcare at home, hospital, or large medical facilities [1], [2]. Instead of being measured face-to-face, with MSNs, patients' health-related parameters can be monitored remotely, continuously, and in real time, and then processed and transferred to medical databases. This medical information is shared among and accessed by various users such as healthcare staff, researchers, government agencies, insurance companies, and patients. Through this way, healthcare processes, such as clinical diagnosis and emergency medical response, will be facilitated and expedited, thereby greatly increasing the efficiency of healthcare.

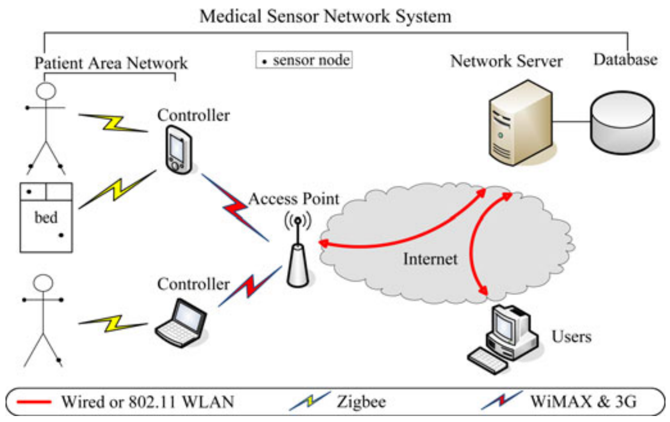

Fig. 1. Architecture of a typical medical sensor network.

Fig. 1 shows the architecture of a typical MSN. A large-scale MSN accommodates tens of patient area networks (PANs). Each PAN consists of some biosensor nodes and a local processing unit (e.g., tablet PC, smartphone, or laptop PC), which is referred to as the *controller*. Here, biosensor nodes (either implantable or wearable) are attached to a patient to periodically collect his/her personal health information (PHI) and forward them to the controller; then, the controller serving as a gateway will report the collected PHI to the network server. Thus, users can issue commands to access the collected PHI or control the biosensors of a target PAN. Note that the sensor nodes communicate only with the corresponding controller, but not individual users. This is because in a large healthcare facility, it would be difficult for the resource-limited sensor nodes to authenticate hundreds of users. As a result, users are authenticated at the controllers (or the network server) which will issue commands to the biosensor nodes on behalf of them.

Since the collected PHI is private information of patients and plays a critical role in medical diagnosis and treatment, it is essential to strictly limit the access of these data to authorized users only in order to ensure the security of these data and preserve the patients' privacy [1].

However, designing a secure and efficient system for MSNs is a difficult task as it faces the following challenges. First, as we will discuss in Section IV, there are quite a number of security requirements that need to be satisfied. Second, the security mechanisms must be resource-efficient since the biosensor nodes, especially the implanted ones, have much lower

Manuscript received November 25, 2012; revised March 13, 2013 and April 26, 2013; accepted June 10, 2013. Date of publication June 17, 2013; date of current version December 31, 2013. This work was supported by the National Natural Science Foundation of China under Grants U1135004 and 61170080, the Guangxi Key Laboratory of Trusted Software, the Fundamental Research Funds for the Central Universities, and Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2011). D. He and S. Tang are co-corresponding authors.

D. He is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China, and also with the Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China (e-mail: hedaojinghit@gmail.com).

S. Chan is with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong (e-mail: eeschan@cityu.edu.hk).

S. Tang is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: csshtang@scut.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JBHI.2013.2268897

2168-2194 © 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

processor speed, memory, transmission speed, and energy supply than sensors of other sensor networks. Thus, as we will discuss in Section II, the limited resources of the biosensor nodes render the use of most common cryptographic techniques (e.g., public key cryptosystems and their enhanced variations such as ring signature) unrealistic or impossible in most circumstances with respect to delay. Moreover, if the message authentication or encryption/decryption methods are not fast enough, an adversary may launch a Denial-of-Service (DoS) attack (e.g., a signature-based DoS attack [3]) to exhaust the resources of the legitimate biosensor nodes and render them less capable of carrying out their intended functions. Third, due to their small size, sensor nodes can be easily stolen or simply lost. Patients' mobility also increases the chances of losing the sensor nodes. Generic wireless sensor networks (WSNs) do not have this problem as they have stationary sensor nodes. Furthermore, the adversary can easily find their targets in local hospitals or healthcare facilities, as opposed to remote locations like forests or battlefields. Therefore, physical compromise of sensor nodes is more likely in MSNs than in generic WSNs.

To address the above challenges, this paper makes three main contributions.

1) We show the security weaknesses and efficiency problems of the existing security systems in MSNs. Then, we identify the characteristics of an MSN and present the requirements of a secure and lightweight system of MSNs. Considering the special features of an MSN, *a powerful mobile adversary* is introduced into MSNs.
2) We propose a secure and lightweight system for MSNs, which not only enables lightweight key management but also provides fine-grained access control in MSNs. In addition, our theoretical analysis demonstrates that the proposed system can meet the requirements.
3) We also implement the proposed system in a network of resource-limited sensor nodes and laptop PCs. Evaluation results show the efficiency of the system in practice. Accordingly, some suggestions on how to set the parameters of the proposed protocols are provided.

The rest of this paper is structured as follows. In Section II, we first survey and analyze the related work and then discuss their security weaknesses and efficiency problems. Section III describes the network and adversary models, and then discusses the features of MSNs. Section IV presents the requirements of a secure system of MSNs. Then, in Section V, we describe our proposed system accompanied by theoretical analysis of the security properties. Section VI describes the implementation and experimental results of our system via real resource-limited sensor platforms. Finally, Section VII concludes this paper.

II. Related Work

Despite the need and importance, to the best of our knowledge, until now no secure and lightweight data transmission and access control platform for MSNs has been proposed. For example, the designers of CodeBlue [4] and MUSIC [5] point out the need for security in a medical environment, but their work does not focus on addressing security issues.

Recently, an architecture called sensor network for assessment of patients (SNAP) [6] has been proposed to address the security challenges facing a sensor network for wireless health monitoring. However, we observe that SNAP does not deal with user authentication for the medical data. Moreover, the collected data from a biosensor are transmitted to the controller in plaintext. Thus, an adversary can easily modify the medical data and/or inject the polluted medical data into the network. Some researchers (see, e.g., [7], [8]) utilize physiological signals (e.g., heart rate interval, blood flow, and electrocardiography) obtained from the patient to enable biosensors to agree upon a symmetric (shared) cryptographic key in an authenticated manner. However, they demand that each biosensor can measure the same physiological parameter; this assumption is rather restrictive and makes this method not suitable for many MSN applications.

Based on the public key cryptography, some novel protocols (see, e.g., [9]–[12]) have been proposed to ensure security of MSNs. The authors of [9] suggest to use the elliptic-curve cryptography (ECC) algorithm to set up symmetric keys between sensor nodes and the base station. Also, a novel group key management and authentication mechanism is presented. However, they are computation-inefficient, cannot fulfill the stringent delay requirements in MSNs, and are vulnerable to DoS attacks. For instance, as reported in [9], the ECC key agreement takes 7.198 s on a Tmote Sky mote, which features a 16-bit, 8-MHz MSP430 processor. Additionally, as described in [10], the elliptic curve Diffie–Hellman (ECDH) key generation used in sensor-to-sensor authentication takes 5.97 s on a Tmote Sky mote. As the common biosensor nodes have less computation power than Tmote Sky motes, public key cryptography is not favorable for the biosensor nodes.

Also, a lightweight identity-based cryptography named IBE-Lite has been proposed [11]. It balances security and privacy with accessibility. However, we observe that there are security weaknesses and efficiency problems in IBE-Lite. First, all the medical data are encrypted by ECC, which is not efficient for MSNs. Second, their work does not consider sensor-to-sink (or user) data authentication. Thus, false medical data could be injected or treated as legitimate due to the lack of node authentication. Third, IBE-Lite cannot resist node replication attacks. That is, an adversary can insert additional hostile biosensors into the network. Fourth, the master key of each PAN consists of n secret keys, which are picked by the patient. Each doctor uses the secret key from the certificate authority to decrypt the messages encrypted by a sensor node. Once a doctor sends n user queries to a target PAN, he/she is able to generate the master key of the PAN. Thus, to ensure the security of IBE-Lite, the number of user queries has to be limited. Le *et al.* [12] presented a mutual authentication and access control protocol, which is based on ECC. A recent study [13] has shown that the scheme is susceptible to information-leakage attacks.

Although there are a lot of works about generic WSNs and mobile ad hoc networks (MANETs) security (see, e.g., [14]–[16]), these mechanisms are not directly applicable in MSNs due to the unique and challenging operational and security requirements of MSNs. For instance, the authors of

[14] introduce a novel approach to ensure distributed privacy-preserving access control, which is built on a ring signature technique. Also, in [15], a self-contained public key-management scheme has been proposed for wireless ad hoc networks, in which a small number of cryptographic keys are stored offline at individual nodes before deployment. Also, to avoid the weaknesses of a public key infrastructure, as a special form of the public key cryptography, identity-based cryptography has been used in various areas of securing MANETs [16]. Unfortunately, as described before, solutions relying on public key cryptography are not directly applicable to MSNs.

III. Network Model, Adversary Model, and Unique Features of MSNs

A. Unique Features of MSNs

MSNs are different from MANETs and WSNs in the following aspects [2].

1) *Data Rate*: Events monitored by MANETs and WSNs usually occur at irregular intervals. On the contrary, MSNs are employed to monitor humans' physiological activities, which more or less may occur periodically. As a result, data streams of applications exhibit relatively stable rates. All nodes are assumed to have loosely synchronized clocks with the help of some existing secure time synchronization scheme.

2) *Mobility*: Relatively, there is no movement between sensors as they are all in the same patient. Movement between controllers and sensors is due to mobility of patients, which is very low.

3) *Efficiency*: The sensed signals can be efficiently processed by biosensors to obtain estimates of physiological information. Also, the power consumption on biosensors is low, and thus, batteries can last longer.

B. Network Model

All the biosensor nodes in an MSN have limited power supply, storage space, and computational capability. Due to the constrained resources, computationally expensive and energy-intensive operations such as the public key cryptography are not favorable for such nodes. We assume that the network server is secure. That is, the network server is equipped with a tamper-resistant component for storing the keying materials. According to the data rate feature of an MSN described in Section III-A, we assume that time is divided into equal and fixed *collection rounds* and each biosensor collects a single data item per round.

The sensor nodes may be placed in, on, or around the patient's body. Although there is no consensus on the communication technologies in PANs, the communication ranges of off-the-shelf technologies (e.g., Zigbee) are larger than 3 m. Thus, according to the mobility feature of an MSN described in Section III-A, we assume that all sensor nodes in a PAN can directly communicate with the controller; thus, a star topology is assumed.

C. Adversary Model

We assume that an adversary can behave as both outside and inside attackers. Outside attackers can drop messages by jamming the communication channel, eavesdrop messages, modify messages, inject forged messages, or replay old messages. Insider attackers can compromise a number of biosensor nodes, controllers, and network users to obtain their data and keying materials.

Considering the special features of an MSN, *a powerful mobile adversary* [17] is introduced into MSNs. One important feature that separates it from other adversary models is its mobility. More specifically, the adversary can compromise different subsets of biosensors in different time intervals. The subset of compromised nodes might not be clustered or contiguous, that is, concurrently compromised nodes can be spread over the entire MSN. While in control of a biosensor node, the adversary acquires keying materials and status, reads all storage/memory, and can eavesdrop on all incoming and outgoing communications of the compromised node. There are two reasons to consider such a mobile adversary model. First, since the biosensors are in, on or around the body of the patient, the compromised biosensors are easily detected by the patient or the health staff. Thus, the adversary roams around the MSN gradually to avoid being detected. Second, it is extremely difficult for the adversary to predict a patient's movement and follow him/her everywhere. As a result, the adversary may inevitably lose control of the already compromised biosensor nodes.

IV. Requirements of a Secure System

In this section, we present several criteria that represent desirable characteristics in a secure and lightweight system for MSNs.

1) *Lightweight*: Every PAN often consists of low-end sensor nodes, which rely on battery energy [1]. Furthermore, emergency situations in an MSN require the capability for fast medical reaction without disabling security functions. For example, secure PAN setup in emergency situations must be carried out in less than 1 s and the maximum allowable latency for electrocardiogram transmission is 250 ms [18]. To match the low capabilities of the sensor nodes, it is important to minimize computation, communication, and storage overhead on the sensors. Hence, cryptographic algorithms must be as fast as possible in order to satisfy these requirements and be invulnerable to DoS attacks.

2) *Fine-grained data access control*: Access control needs to be enforced for the patient-related data in the whole MSN so that private information will not be obtained by unauthorized users. More importantly, a secure system should provide different privileges for different network users.

3) *Scalability*: The system should be efficient even in a large scale MSN with many users and many PANs [1].

4) *Flexibility*: The access policy should be adapted dynamically to contexts, such as time, location, or certain events related to patients. Note that in MSNs, the access policy should be defined by both patients and healthcare units. For example, on-demand authorization to read a patient's PHI can be given temporarily to an available doctor who is not on the access list when a medical emergency happens. Obviously, inability or irresponsiveness in adapting the access rules may threaten a patient's life [1].

5) *Confidentiality*: In order to prevent the patient-related data from leaking, the data need to always be kept confidential at a node or local server (i.e., the network server). Data confidentiality should be resistant to device compromise attacks (e.g., node compromised and controller compromised attacks). That is, compromising one node helps the adversary to gain nothing or little from the data stored at that node.

6) *Data integrity assurance*: In MSNs, the patient-related data are vital, and modified data would lead to disastrous consequences. Therefore, data integrity shall be protected all the time.

7) *Forward secrecy*: It means that even if an adversary obtains the current secrets of a node, it cannot decrypt (or forge authentication tags for) those data collected and encrypted (or authenticated) before compromise.

8) *Backward secrecy*: It means that even an adversary has compromised (and then released) a node, it cannot decrypt (or forge authentication tags for) those data collected and encrypted (or authenticated) by the node after releasing.

9) *Strong contextual privacy preservation*: We divide privacy issues in MSNs into content-oriented privacy and contextual privacy. Here we just focus on contextual privacy, since the content-oriented privacy has been considered in Requirements 5, 7, and 8. Contextual privacy means an adversary has the ability to link the source and the destination of a message. In an MSN, if an adversary can link the patient with a specific physician, then the patient's privacy will be lost. Thus, it is very important to protect contextual privacy, which includes sensor identity privacy and PAN identity privacy of every collected data, and each user's privilege content privacy in addition to the privacy of every user command content. For example, if an adversary searches the whole MSN for a specific parameter, protecting the privacy of sensor identity of every collected data is desirable. Similarly, if an adversary searches the whole MSN for a specific patient, protecting the privacy of PAN identity of every collected data is desirable. In addition, we illustrate the importance of user privilege content privacy by considering the following two scenarios. One is that often each user's privilege content indicates the user identity information and the relation between the user and some patients (i.e., the owner of some PAN), thus exposing the patients' privacy. The other is that with the knowledge of some user privileges, an adversary can seek the important users and then launch attacks on the MSNs.

V. Proposed System

A. Basic Idea of Our System

The basic idea of the proposed system is given as follows. After a user registers to the network server, he/she is allowed to issue commands to access the collected PHI or control the biosensors according to his/her privilege. To achieve this goal, proxy-protected signature by warrant (PSW) [19] is introduced into our system. This technique is a special digital signature. There are two kinds of participants, i.e., an original signer and proxy signers. The original signer gives the proxy signer a warrant, which specifies the identity of the proxy signer, the identity of the original signer, the expiration time of the delegation of signing power, etc. The proxy signer generates proxy signatures only with the proxy signature key given by the original signer. Verifiers validate proxy signatures only with the public key of the original signer and pay attention to the legality of the warrant.

The detailed information about applying the PSW technique into the proposed system is given as follows. The network server of an MSN plays the role of the original signer, while the users of the MSN play the role of proxy signers. Through registration, the users obtain one or more proxy signature keys from the network server before they enter to an MSN. The key can subsequently be used to make signature on a command. Thus, authorized users generate valid commands only with the proxy signature keys given by the network server. The validity of each command can be verified by the controller of any PAN or the network server with the public key. Through this way, the network server can prevent unauthorized commands on the MSN. Additionally, our proposed system only requires lightweight cryptographic operations (i.e., symmetric-key encryption/decryption and hash function operation) and do not need verification tables to be stored on a biosensor. Hence the computational and storage requirement on a biosensor is low.

Quite a number of PSW schemes have been proposed in the literature. However, some of them suffer from some security weaknesses, and most of them are not efficient enough for biosensors. After a thorough evaluation, we have found that Shao's PSW scheme [20] is most suitable for our purpose. However, in spite of its efficiency, we observe that this scheme has a design weakness, which will cause failure of the proposed protocol. Thus, to ensure that the proposed protocol works, a feasible approach has been proposed to fix such a weakness.

Our system involves four phases. The system initialization phase is performed by the network server to set up an MSN. User joining phase is involved before a user can issue commands to the MSN. During the regular use phase, the data from each biosensor node are securely transmitted to the network server via the controller. In the user command phase, if a network user has a new command, he/she will need to construct the command and the proxy signature and then send them to the network server (or the controller of a target PAN). If the command verification passes, the network server (or the controller of a target PAN) responds to the user's command. Fig. 2 illustrates the flows of security information of the proposed system. More detailed description will be provided in the following sections.

B. System Structure

1) System Initialization Phase: In this phase, the network server executes the following steps. The notations used throughout this paper are listed in Table I.

a) Randomly pick two large safe primes p and q (i.e., primes p and q such that $(p-1)/2$ and $(q-1)/2$ are primes as well), and compute a public modulus $n = pq$. Then the server chooses a public one-way hash function $h()$ such as SHA-1.

b) Choose a pair of integers e and d satisfying the properties $e{\cdot}d{\equiv}1\,(\bmod\,\phi(n))$ and d is a large positive number, where $\phi(n) = (p-1)(q-1)$ is Euler's totient function and

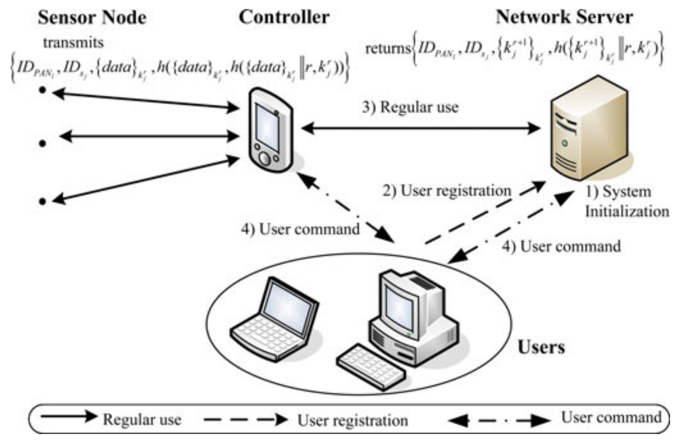
Sensor Node
Controller
Network Server
transmits
$\{ID_{PAN_l}, ID_{s_j}, \{data\}_{k_j^r}, h(\{data\}_{k_j^r}, h(\{data\}_{k_j^r} \| r, k_j^r))\}$
returns $\{ID_{PAN_l}, ID_{s_j}, \{k_j^{r+1}\}_{k_j^r}, h(\{k_j^{r+1}\}_{k_j^r} \| r, k_j^r)\}$
3) Regular use
2) User registration
1) System Initialization
4) User command
4) User command
Users
Regular use
User registration
User command

Fig. 2. Flows of security information in the proposed system.

TABLE I
NOTATIONS

Notation	Descriptions
U_i	the ith user
S_j	the jth sensor node
r	round index
SK_{tns}	the private key of the network server
PK_{tns}	the public key of the network server
$(X)_K$	encrypting message X with a symmetric key K
, or $\|$	concatenation operator
ID_A	the identity of an entity A
$h(.)$	a public one-way collision-resistant hash function
$h(X, K)$	keyed hash function with a session key K for message X

e should be larger than the output of the one-way hash function $h()$. Thus, based on the Rivest, Shamir, and Adleman algorithm, the network server creates its public and private keys as $PK_{tns} = \{n, e\}$ and $SK_{tns} = d$, respectively.

c) For every PAN, say PAN_l, the network server secretly shares the distinct initial key k_j^0 with every biosensor, say s_j, and stores the tuple $\langle ID_{PAN_l}, ID_{s_j}, k_j^0 \rangle$. Here ID_{s_j} is the identity of the node s_j while ID_{PAN_l} is the identity of the PAN PAN_l. In general, the maximum number of the biosensor nodes in a PAN is less than 40; thus, the bit length of ID_{s_j} is set to 8. Also, the maximum number of the PANs in every MSN is less than 200; thus, the bit length of ID_{PAN_l} is set to 8. There are two suitable general initial key distribution schemes: trusted-server schemes and key predistribution schemes. Trusted-server schemes depend on a trusted server for key agreement between the network server and each node. Different from generic WSNs, this type of scheme is suitable for MSNs. This is because for an MSN, a trusted infrastructure is often in place. The second way, where key information is distributed to each node prior to deployment, has been widely employed in many generic WSNs. Compared to generic WSNs, this approach is more feasible for MSNs. The reason is that because the scale of an MSN is much smaller than that of a generic WSN, this scheme has little impact on the efficiency requirements such as flexibility. No matter which of these two schemes is used, the initial distribution scheme is only executed in the system initiation phase, it has small impact on the complexity of the proposed system. Also, the parameter PK_{tns} is loaded into the controller of each PAN before the network deployment.

2) User Joining Phase: Before a user, say U_i, can issue commands to the MSN, he/she needs to register to the network server. After verifying his/her registration information, the network server assigns an identity, say ID_{U_i}, for him/her. We assume that the length of each user identity is 2 bytes, in which case the system can support 65 536 network users. Then, the network server computes a proxy signature key υ_i for user U_i which is given by

$$\upsilon_i \equiv [h(m_w)]^{-d} \pmod n. \qquad (1)$$

Here, n is the public modulus defined earlier. The warrant m_w records ID_{U_i}, the identity of the network server and the user privilege such as the identities of PANs that user U_i is allowed to access, and valid periods of delegation (i.e., the beginning time and the end time).

Note that (1) involves modular exponentiation with a negative exponent, which can be performed by finding the multiplicative inverse u of $h(m_w)$ modulo n using the extended Euclidean algorithm. That is, $\upsilon_i \equiv [h(m_w)]^{-d} \pmod n \equiv u^d \pmod n$, where $h(m_w) \cdot u \equiv 1 \pmod n$. It should be noted that u exists if and only if $h(m_w)$ and n are coprime [i.e., $\gcd(h(m_w), n) = 1$]. However, as the PSW scheme of [20] does not require $h(m_w)$ and n to be coprime, this scheme and the proposed system fail when $h(m_w)$ and n are not coprime. This will happen when p or q divides $h(m_w)$. To ensure that the proposed protocol works, a feasible approach is when the network server computes m_w for a user, redundant bits are appended into m_w such that $h(m_w)$ and n are coprime [3]. Of course, instead of adding redundant bits, a simple and efficient solution is to set p and q to be larger than $h(m_w)$ so that they cannot divide $h(m_w)$, and therefore, $h(m_w)$ and n are coprime. For example, when SHA-1 function is used, the length of $h(m_w)$ is 160 bits. If the length of n is 1024 bits, the lengths of p and q are set to 512 bits, respectively.

For user U_i, the network server returns the message $\{\upsilon_i, m_w\}$ in a secure manner (e.g., using the wired transport layer security protocol). Afterward, user U_i can verify the proxy signature key υ_i by checking if (2) is satisfied:

$$\upsilon_i{}^e h(m_w) \equiv 1 \pmod n. \qquad (2)$$

Because

$$\upsilon_i \equiv u^d \pmod n, \;\; u \equiv \upsilon_i{}^e \pmod n, \;\; u \equiv h(m_w)^{-1} \pmod n,$$
$$\upsilon_i{}^e \pmod n \equiv h(m_w)^{-1} \pmod n.$$

3) Regular Use Phase: After the system initiation phase, the MSN is ready for regular use. For simplicity, in the following we consider PAN PAN_l as an example.

At the end of each round $r \geq 0$, the complete message that node s_j sends to the network server through the controller is

$$ID_{PAN_l}, ID_{s_j}, \{data\}_{k_j^r}, h(\{data\}_{k_j^r} \| r, k_j^r) \qquad (3)$$

where $data$ is the collected data unit by node s_j during round r, and $\{ID_{PAN_l}, ID_{s_j}\}$ denotes the source address of the message. After that, node s_j generates a secret key by $k_j^{r+1} = h(k_j^r)$, and promptly erases k_j^r from its memory. Data transmission is a costly operation in wireless networks; sending one bit over a wireless medium requires over 1000 times more energy than a single 32-bit computation [21]. In order to reduce the transmission overhead, for (3), $h()$ is a secure hash function (e.g., SHA-1) with truncated output (e.g., say of 10 bytes). Due to the limitation of the storage resource on each controller, the controller needs to submit the collected data to the network server for permanent records. Upon receiving the message, the network server retrieves the shared key k_j^r according to the received information $\{ID_{PAN_l}, ID_{s_j}\}$. Thus, the network server can use the key k_j^r to verify the authenticity of the sender, and the freshness (through checking the round index r) and integrity (through checking the hash value) of the message, and obtain the data unit $data$. Subsequently, the network server computes the next key $k_j^{r+1} = h(k_j^r)$, and then replaces k_j^r of the tuple $\langle ID_{PAN_l}, ID_{s_j}, k_j^r\rangle$ with k_j^{r+1}.

From (3), we can see that the PHI encryption key for each round is never reused, thus minimizing the risk of key discovery attacks. This leaves the adversary the only choice of brute-force attacks. Since keys are hash values, dictionary attacks do not apply. With a reasonable length of hash values, such as 160 bits coupled with a strong encryption algorithm, it will be very difficult for the adversary to crack the keys.

4) User Command Phase: As described in Section V-B, after obtaining proxy signature keys, if a user, say U_i, wants to issue a command, he/she will need to construct the command Que and then make a signature on Que as follows.

a) Choose a random integer $z \in [1, n]$ and compute $\beta = z^e \pmod n$.
b) Use the command Que to compute $\delta = h(Que\|\beta)$. Note that to defeat the replay attack, user U_i needs to attach a timestamp T_i into Que.
c) Compute $y = z \times {v_i}^{\delta} \pmod n$. Finally, U_i sends $\{Que, m_w, y, \delta\}$ to the network server (or the controller of a target PAN).

Upon receiving a signature message $\{Que, m_w, y, \delta\}$, the network server (or the controller) verifies it by carrying out the following operations. Here, we consider the network server as an example.

a) Check whether the timestamp T_i included in Que is within some allowable range compared with the current time. If the result is negative, the signature message is rejected; otherwise, the network server pays attention to the legitimacy of the warrant m_w and the command Que. For example, according to the valid periods of delegation field of warrant m_w, the network server can check whether the privilege of a user has expired. Another example is that the network server checks whether the command Que is within the scope of the user privilege by the definition of warrant m_w. Only if they are valid, the verification procedure goes to the next step.

b) Compute $\beta^* = y^e h(m_w)^{\delta} \pmod n$. c) Check whether $h(Que\|\beta^*) = \delta$. Because

$$v_i^e = h(m_w)^{-1} \pmod n$$
$$\beta^* = z^e v_i^{\delta e} h(m_w)^{\delta} = z^e = \beta \pmod n$$

Thus,

$$h(Que\|\beta^*) = h(Que\|\beta) = \delta$$

If all verification procedures described previously pass, the network server (or the controller) believes that the command Que and the warrant m_w are from an authorized user with the necessary privileges. The user can interact with the network server (or the controller of a target PAN) either directly or remotely (e.g., from the user's home) for ease of patient monitoring. Commands issued from a network user may activate the patient's biosensors or adjust their sampling frequency and other parameters. Such commands will be forwarded by the network server to the specified biosensor. Alternatively, if the network user sends an access command, the network server returns the resulting PHI to the user. For security reason, the user can establish a session key with the network server through some way (e.g., ECDH key exchange) and use the session key to ensure the confidentiality, integrity, and freshness of PHI transmission.

5) Security Analysis: In the following, we will analyze the security of the system to verify whether the security requirements mentioned in Section IV have been satisfied.

Lightweight and scalability: The system only requires lightweight cryptographic operations (i.e., symmetric-key and hash function operations) on a biosensor. Moreover, considering the memory constraint on the biosensor nodes, only an updated symmetric key is stored in each node. Hence, the computational and storage requirement on a biosensor is low. Moreover, the computation complexity on the biosensor nodes (or network users) is independent of the total number of biosensor nodes or network users. Thus, the system is scalable.

Fine-grained data access control, flexibility and data integrity assurance: As described in Section V-B2 and V-B4), in order to pass the signature verification of each controller and the network server, each network user has to register to the network server; then, the network server assigns him/her a proxy signature key according his/her user privilege. The network server can restrict each network user's privilege by the definition of the warrant. Therefore, the network server enforces fine-grained data access control by user registrations. It is clear that the access policy can be defined by both patients and the healthcare units. Therefore, the system can provide fine-grained data access control flexibly. Also, as described in Section V-B3), in order to provide data integrity assurance, a keyed hash function has been used, i.e., $\{\{data\}_{k_j^r}, h(\{data\}_{k_j^r}\|r, k_j^r)\}$. Upon receiving such a message, the network server checks the validity of $h(\{data\}_{k_j^r}\|r, k_j^r)$. If the result is positive, the network server believes this message is not altered; otherwise, the network server simply drops the message.

Confidentiality and forward secrecy: The system updates the encryption key at each round using the one-way hash function. Such keys are used to realize forward-secure encryption of data produced in each round. For example, we assume that an adversary compromises node s_j at round $r+1$. In this case, the adversary will not be able to read the encrypted PHI submitted by node s_j in the prior r rounds, even if it compromises all controllers and any other nodes. This is because all the encryption/decryption keys $\{K_j^x\}$, $0 \leq x \leq r$, no longer exist in node s_j.

According to the above analysis, the system meets Requirements 1–7 described in Section IV. However, the system can not satisfy Requirement 8. That is, it cannot provide backward secrecy yet. This is addressed in the next section.

C. Backward Secrecy Extension

Ensuring backward secrecy requires some changes to the proposed system. For brevity, we just present the parts that need to be changed.

1) Regular Use Phase: At round $r \geq 0$, every sensor node, say s_j, transmits $\{ID_{PAN_l}, ID_{s_j}, \{data\}_{k_j^r}, h(\{data\}_{k_j^r} \| r, k_j^r)\}$ to the network server through the corresponding controller. Upon receiving this message, the network server uses the corresponding key k_j^r to check the validity of the sensor data. If it is valid, the network server picks a random number k_j^{r+1} as the next key. After that, the network server generates the response $\{ID_{PAN_l}, ID_{s_j}, \{k_j^{r+1}\}_{k_j^r}, h(\{k_j^{r+1}\}_{k_j^r} \| r, k_j^r)\}$ and then sends it to node s_j through the controller, where $\{ID_{PAN_l}, ID_{s_j}\}$ denotes the destination address of this message. At the same time, the network server replaces the key k_j^r of the tuple $\langle ID_{PAN_l}, ID_{s_j}, k_j^r \rangle$ with k_j^{r+1}. Upon receiving this message, with the received information $\{k_j^{r+1}\}_{k_j^r}$, node s_j uses the key k_j^r to compute $h(\{k_j^{r+1}\}_{k_j^r} \| r, k_j^r)$ in order to verify the authenticity of the sender, and the freshness and integrity of the message. If the result is positive, node s_j decrypts the information $\{k_j^{r+1}\}_{k_j^r}$ and obtains the next key k_j^{r+1} and then erases the old key k_j^r. At the same time, with such a response from the network server, the controller (and node s_j) believes that the network server has received the sensor data during round r.

2) Security Analysis: Here, it is demonstrated that this extension can provide backward secrecy, i.e., Requirement 8. As described in Section III-A, the network server is tamper-proof and securely picks a random number as the encryption/decryption key of each round for each biosensor node; thus, the adversary cannot obtain the secret key. Referring to Requirement 8 described in Section IV, suppose the adversary compromises node s_j at round r obtains the key k_j^r and then releases node s_j at round $r + 1$. That is, the adversary does not obtain the response $\{ID_{PAN_l}, ID_{s_j}, \{k_j^{r+1}\}_{k_j^r}, h(\{k_j^{r+1}\}_{k_j^r} \| r, k_j^r)\}$. Obviously, the adversary cannot generate the new key k_j^{r+1}, even if it compromises all controllers and any other nodes.

D. Contextual Privacy Preservation Extension

Note that the current system cannot provide strong contextual privacy preservation, i.e., Requirement 9 is not satisfied yet. More exactly, there are four issues about this requirement exist in the system as follows.

1) *Command exposure*: The commands from users are transmitted in plaintext in free air.
2) *Warrant exposure*: The warrant from a user, which indicates the user privilege, is transmitted in plaintext in open environment.
3) *Sensor node identity exposure*: The sensor node ID, which is attached to the sensor data, is transmitted in plaintext.
4) *PAN identity exposure*: The PAN ID is transmitted in plaintext.

To avoid command exposure and warrant exposure, a simple improvement for the system is as follows.

When a user, say U_i, hopes to issue a command to the network server (respectively, the controller of a target PAN), the command is encrypted with the network server's public key (respectively, the controller's public key). At the same time, the warrant is encrypted in the same way.

To avoid sensor node identity exposure, we propose a hashing-based identity randomization approach, which can be directly employed in the system. The basic idea is for each sensor node, say s_j, to use a one-way keyed hash function for producing a hash value as its identity. As one of the inputs of a keyed hash function, the key k_j^r introduced in the backward secrecy extension can be used. At the same time, the original identity of the sensor node, which is assigned by the network server before the network deployment, is used to be the other input of a keyed hash function. As mentioned previously, the key k_j^r at round r is updated for every round between the sensor node s_j and the network server. Thus, this approach can evolve sensor node identities such that it can provide forward secrecy and backward secrecy. The adversary cannot link two values as the input and output of a keyed hash function without knowing the key k_j^r. Without knowing the keys, the adversary cannot determine whether two messages are sent by the same sensor node through eavesdropping. Therefore, the anonymity of a message's sender is protected as long as the key is not compromised. A sensor node can erase its current ID and the key k_j^r, and then generate a new identity after receiving the response from the network server (more exactly, obtaining the new key k_j^{r+1}). Then through physically capturing a sensor node, the adversary only obtains the original ID of the sensor node. Because of one-way feature of the keyed hashing, the adversary cannot reverse the hash function to get the previously used IDs. Clearly, the above approach can also be employed easily to avoid PAN identity exposure. This requires more changes to the system. For brevity, we just present the parts that need to be changed.

1) System Initiation Phase: With initial key distribution scheme described previously, the network server secretly shares a random number k_l^0 as the distinct initial key with each PAN, say PAN_l, and stores the tuple $\langle ID_{PAN_l}, k_l^0 \rangle$. The key k_l^0 is loaded into each biosensor node and the controller of PAN PAN_l before the network deployment.

2) Regular Use Phase: At round $r \geq 0$, the complete message from node s_j to the corresponding controller are of the form $M = \{H1, H2, r, \{data\}_{k_j^r}, h(\{data\}_{k_j^r} \| r, k_j^r)\}$. Here $H2 = h(ID_{PAN_l}, k_l^r)$ is a keyed hash value that identifies the receiver of this message. In addition, $H1 = h(ID_{s_j}, k_j^r)$ is a hash value that identifies the original message sender. That is, $H1$ and $H2$ denote the source and destination addresses of this message, respectively. In order to reduce the transmission overhead, for the generation of $H1$ and $H2$, $h()$ is a hash function with truncated output of, say, 3–10 bytes. After that, node s_j generates a secret key by $k_l^{r+1} = h(k_l^r)$, and promptly erases k_l^r from its memory. Because of the shared medium of wireless networks, when node s_j is sending messages, all the nodes that are in its radio range can receive the message. To preserve the receiver's identity anonymity, $H2 = h(ID_{PAN_l}, k_j^r)$ is used to notify the real receiver. The controller receives a message from one of its sensor nodes, but initially does not know the sender and receiver of this message since the identity of the receiver is hidden in $h(ID_{PAN_l}, k_j^r)$ and the identity of the sender is hidden in $h(ID_{S_j}, k_j^r)$. The round number r included in the message allows the controller to interpret $H2(=h(ID_{PAN_l}, k_l^r))$ correctly. Note that because of the properties of one-way keyed hash functions, the controller cannot simply compute an inverse hash function to get the original identity of the receiver directly. However, because the controller has the current secret key k_l^r, it uses the key k_l^r to compute $h(ID_{PAN_l}, k_l^r)$ and then compares it with the received $H2$. If the result is positive, the controller transmits the message M to the network server; otherwise, the controller simply drops the message. Note that without knowing the secret k_j^r, the controller does not know the sender of this message. In addition, upon receiving the message M, the network server initially does not know the sender of this message, since the identity of the sender is hidden in $\{H1, H2\}$. As described previously, the round index r included in the message allows the network server to interpret $\{H1, H2\}$ correctly. Again, because of the properties of one-way keyed hash functions, the network server cannot simply compute an inverse hash function to get the original identity of the sender directly. However, it can use the secret keys $\{k_j^r, k_l^r\}$ from the tuples $\langle ID_{PAN_l}, k_l^r\rangle$ and $\langle ID_{PAN_l}, ID_{s_j}, k_j^r\rangle$ to check the validity of $\{H1, H2\}$ and then find which biosensor node is the sender of this message. At the end of round r, the network server generates the secret key $k_l^{r+1} = h(k_l^r)$ for PAN PAN_l, and then replaces the key k_l^r of the tuple $\langle ID_{PAN_l}, k_l^r\rangle$ with k_l^{r+1}. Also, the controller of PAN PAN_l generates the secret key $k_l^{r+1} = h(k_l^r)$, and promptly erases k_l^r from its memory.

3) Security Analysis: Here it is demonstrated that the system with this extension can satisfy Requirement 9. The command (or the warrant) is encrypted with the network server's public key (respectively, the controller's public key), thus, only the network server (respectively, the controller) can use the corresponding private key to obtain the command (or the warrant). As described previously, the hashing-based identity randomization method is proposed to avoid sensor node identity and PAN identity exposure.

VI. Implementation and Performance Evaluation

We evaluate the proposed system by implementing all components on an experimental testbed.

A. Implementation and Experimental Setup

In order to investigate the feasibility of the proposed system on the biosensor nodes, same as the existing studies [4]–[11] on securing MSNs, we choose two common resource-limited sensor nodes, i.e., TelosB and MicaZ motes. The TelosB mote is equipped with an 16-bit, 8-MHz MSP430 microcontroller, 10-kB RAM, 48-kB ROM, 1024-kB flash and an 802.15.4/ZigBee radio. Also, the MicaZ mote features an 8-bit, 8-MHz Atmel microcontroller with 4-kB RAM and 128-kB ROM. Our implementation has the network server, controller, network user, and sensor node side programs. The protocols operated by the first three entities have been implemented in C (using OpenSSL [22]) and executed in laptop PCs (with 2-GB RAM) under Ubuntu 11.04 environment with different computational power. In addition, the sensor node side programs are written in nesC. Our motes run TinyOS [23] 2.x. Throughout this paper, unless otherwise stated, all experiments on laptop PCs and sensor nodes were repeated one thousand times for each measurement in order to obtain accurate average results. Additionally, for one-way hash function $h()$, we have selected SHA-1, thus the output size is 160 bits.

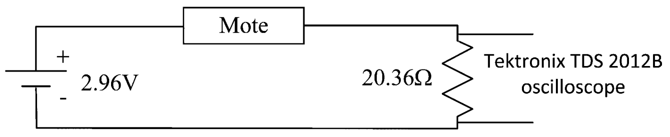

Fig. 3. Experimental setup for investigating the energy consumption.

To measure the power consumption of various cryptographic functions in the sensor nodes, the circuit as shown in Fig. 3 has been built [24]. When the mote in the circuit is executing a cryptographic function, the voltage across the resistor V_r is measured by a Tektronix TDS 2012B oscilloscope. From V_r, the current through the circuit I can be obtained by Ohm's law. Also, the voltage across the mote V_m can be obtained by Kirchoff's voltage law. Then, the power consumed by the cryptographic function is simply given by $V_m \times I$. Moreover, if we measure the execution time of the cryptographic function, we can obtain its energy consumption.

B. Evaluation Results

The proposed system is evaluated according to the following metrics, namely, memory overhead, execution time, and energy overhead. The memory overhead refers to the amount of RAM and ROM space occupied by the protocol codes.

Table II gives the execution time of the main operations in the proposed system when the length of message Que is set to 8 bytes and the length of the parameter modulus n varies. Here, system initialization indicates the generation of the network server's public key and private key. It can be seen that only this operation takes relatively longer time. The execution times of other operations are only of the order of milliseconds. For example, the execution time of proxy signing a message and the signature verification are 1.57 and 1.72 ms for a network user and a controller (or the network server) on a 1.6-GHz laptop PC when the length of n is 1024 bits, respectively. Considering the clock frequency of a typical smartphone is more than 1 GHz, our protocols are efficient for most of local processing units (e.g., smartphone or laptop PC).

Fig. 4 shows the execution time of SHA-1 hash function (extracted from TinyECC 2.0 [25]) on MicaZ and TelosB motes. The inputs to the hash function are randomly generated numbers with length varying from 30 to 198 bytes in increments of

TABLE II
EXECUTION TIME FOR THE MAIN OPERATIONS OF OUR SYSTEM

	Experiment 1			Experiment 2			Experiment 3			Experiment 4		
processor speed (GHz)	1.6			2.2			2.6			3.1		
The length of n (bits)	2048	1024	768	2048	1024	768	2048	1024	768	2048	1024	768
System initialization (ms)	37720.93	2251.7	759.82	27223.76	1603	554.87	23098.22	1386.55	471.69	20325.9	1146.11	387.93
Proxy signature key generation (ms)	33.6	4.66	2.3	25.64	3.49	1.71	21.46	3.09	1.13	17.98	2.37	1.08
Proxy signature key verification (ms)	3.22	0.8	0.56	2.54	0.72	0.44	2.18	0.57	0.39	1.89	0.51	0.31
Proxy signing (ms)	6.11	1.57	1.13	4.14	1.13	0.83	3.77	0.93	0.72	3.32	0.81	0.58
Signature verification (ms)	6.23	1.72	1	4.84	1.14	0.61	4.14	1.06	0.55	3.05	0.87	0.4

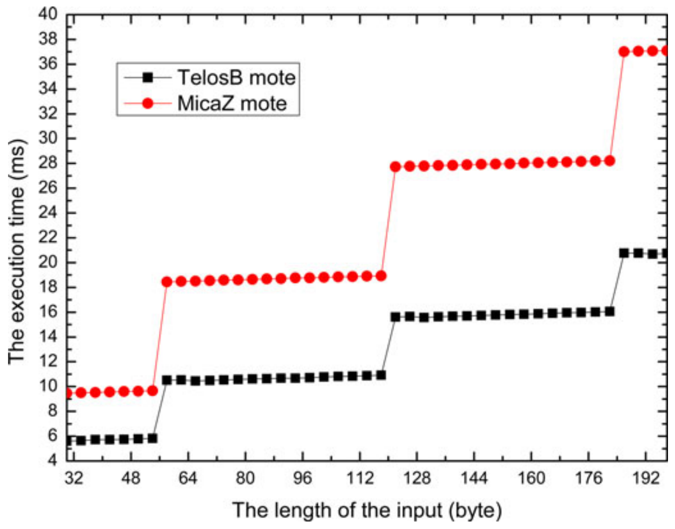

Fig. 4. Execution time of SHA-1 hash function on MicaZ and TelosB motes.

4 bytes. We perform the same experiment ten thousand times and take an average over them. For example, the execution time on a MicaZ mote for inputs of 30, 58, 122, and 186 bytes are 9.4609, 18.4498, 27.7263, and 37.0021 ms, respectively. Also, the execution time on a TelosB mote for inputs of 30 and 58 bytes are 5.6012 and 10.50473 ms, respectively. From Fig. 4, it can be seen that the execution time remains very stable when the byte length of the input falls in the interval $[0, 55]$, $[56, 119]$, $[120, 183]$, or $[184, 198]$. As the input of a hash function, the longer the length of the PHI (i.e.,$\{\{data\}_{k_j^r} \| r \| k_j^r\}$ in the proposed system) in each round, the bigger the time consumption of hash operation. Thus, it is suggested that in our system, the length of the PHI in each round should be chosen according to the above intervals to achieve a balance between the packet length and computing complexity. For example, when the length of the PHI is a bit longer than 55 bytes, according to the system configuration, only 55 bytes data are picked from the PHI as the input of a hash function.

Next, the energy consumption of SHA-1 hash function operation is investigated. When a MicaZ mote is used in the circuit, $V_r = 144$ mV, $I = 7.0727$ mA, $V_m = 2.8160$ V, and $P = 19.9167$ mW. When a TelosB mote is used, $V_r = 40$ mV, $I = 1.9646$ mA, $V_m = 2.9200$ V, and $P = 5.7366$ mW. By multiplying the power with the execution time obtained from Fig. 4, we can determine the total energy consumption of SHA-1 operation on the motes. For example, the energy consumption of SHA-1 operation with a random 30-byte number as input on MicaZ and TelosB motes are 0.1884 and 0.0321 mJ, respectively.

Compared with other commonly known symmetric encryption/decryption algorithms (e.g., RC5, Data Encryption Standard, Triple-DES), the Advanced Encryption Standard (AES) algorithm is regarded as one of the most efficient ones. Moreover, AES is supported by many hardware platforms for the MSNs. We have thus implemented software AES encryption/decryption [26] and CC2420 radio based stand-alone hardware AES encryption on MicaZ and TelosB motes. These two modules are used with ten rounds (with a 128-bit key and 128-bit block size) and the ECB mode. For CC2420 radios hardware AES, the stand-alone mode does not have decryption phase. Table III shows the execution time of software AES encryption/decryption for MicaZ and TelosB motes with the plaintext (i.e., the data item $data$ in this paper) of different lengths, respectively. Also, Table IV shows the execution time of stand-alone hardware AES encryption for MicaZ and TelosB motes with the plaintext of different lengths, respectively. As expected, it can be seen from Tables III and IV that hardware AES operation is much faster than software AES operation for the same level of security. Also, the time consumed by hardware AES implementation is extremely small. For example, even with 160-byte plaintext as input, the encryption procedure takes 0.346 and 0.873 ms on MicaZ and TelosB motes, respectively.

When software AES algorithm is executed in a MicaZ mote, $V_r = 136$ mV and $P = 18.8638$ mW. For a TelosB mote, $V_r = 41$ mV and $P = 5.8783$ mW. By multiplying the power with the execution time given by Table III, we can determine the total energy consumption of software AES encryption/decryption operation on MicaZ and TelosB motes, respectively. For example, the energy consumption of software AES encryption operation with a random 48-byte plaintext as input on a MicaZ mote is 0.0633 mJ. Then the energy consumption of stand-alone hardware AES encryption operation is investigated. When a MicaZ mote is used in the circuit, $V_r = 436$ mV and $P = 54.0502$ mW. When a TelosB mote is used, $V_r = 336$ mV and $P = 43.3036$ mW. For example, the energy consumption of stand-alone hardware AES encryption operation with a random 48-byte plaintext as input on a MicaZ mote is 0.0058 mJ. Thus, the energy consumption of hardware AES is much lower than that of software AES for the same level of security. According to the above analysis, the proposed mechanisms are much more efficient than the public key based protocols (see, e.g., [9]–[12]) in terms of computation complexity and energy consumption.

TABLE III
EXECUTION TIME OF SOFTWARE AES FOR MICAZ AND TELOSB MOTES WITH THE PLAINTEXT OF DIFFERENT LENGTHS

	MicaZ						TelosB					
Plaintext length (bytes)	48	96	112	128	144	160	48	96	112	128	144	160
Encryption time (ms)	3.355	6.829	7.964	9.106	10.238	11.367	5.46	10.864	12.676	14.491	16.287	18.079
Decryption time (ms)	4.001	7.99	9.319	10.644	11.983	13.314	6.517	13.006	15.171	17.343	19.510	21.673

TABLE IV
EXECUTION TIME OF STAND-ALONE HARDWARE AES FOR MICAZ AND TELOSB MOTES WITH THE PLAINTEXT OF DIFFERENT LENGTHS

	MicaZ						TelosB					
Plaintext length (bytes)	48	96	112	128	144	160	48	96	112	128	144	160
Encryption time (ms)	0.108	0.206	0.248	0.275	0.310	0.346	0.278	0.537	0.621	0.704	0.791	0.873

TABLE V
CODE SIZES OF THE WHOLE SYSTEM

		Controller	Node
TelosB	ROM (bytes)	17,058	16,172
	RAM (bytes)	688	463
MicaZ	ROM (bytes)	18,016	15,044
	RAM (bytes)	645	483

Next, we have implemented the proposed system as a whole, where the controller and node side programs are executed on the resource-limited sensor nodes, and the network server side programs are executed on PCs. Each node uses CC2420 stand-alone hardware AES encryption module to encrypt the sensed data. In our implementation, the controller of a PAN communicates with the network server through a serial port. Also, each node uses software AES decryption module to decrypt the response from the network server. Because the key size of the used AES algorithms is 16 bytes while the output of SHA-1 function is 20 bytes, only the first 16-byte data are used in the key update operation. Table V shows the code sizes of the whole system. For example, the implementation of the node side programs on a TelosB mote occupies 16 172 bytes of ROM and 463 bytes of RAM, respectively. The resulting size of this implementation corresponds to only 32.90% and 4.52% of the ROM and RAM capacities of TelosB, respectively. Note that a biosensor node in an MSN is mainly used to collect and transmit the sensor data and does not need to carry out any data processing task. The code sizes of these functions are very small, and hence, the ROM of a biosensor node is only lightly occupied. Therefore, the implementation overhead of 32.90% of ROM is acceptable.

Finally, we compare the total energy of two cases with respect to communication, i.e., a biosensor node sending data to the controller 1) without any security protection scheme; and 2) with our security protection scheme. According to (3), the communication overhead incurred by our security protocol for each round of data transmission is 20 bytes of message authentication code if SHA-1 is used. Assuming that the transmission data rate of a mote is 160 kb/s (the maximum data rate of MicaZ and TelosB motes is 250 kb/s), the transmission time of 20 bytes is 0.977 ms. Given that the C2420 transceiver radio (used on many sensor motes such as MicaZ and TelosB) draws 17.4 mA in transmitting mode [27] and a mote is operated by a 3-V battery, this extra consumed energy is acceptable.

VII. CONCLUSION

In this paper, we have identified the security challenges facing an MSN for wireless health monitoring and then proposed a novel and lightweight system to achieve secure data transmission and access control for MSNs. The security analysis has demonstrated that our system can achieve the requirements of the protocol of this kind. We have implemented the protocols on real mobile devices and sensor platforms with limited-resource. Experimental results have shown that our approaches are feasible for real-world applications.

REFERENCES

[1] K. Lorincz, D. Malan, T. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, S. Moulton, and M. Welsh, "Sensor networks for emergency response: Challenges and opportunities," *IEEE Pervas. Comput.*, vol. 3, no. 4, pp. 16–23, Oct. 2004.

[2] J. Choi, B. Ahmed, and R. Gutierrez-Osuna, "Development and evaluation of an ambulatory stress monitor based on wearable sensors," *IEEE Trans. Inf. Technol. Biomed.*, vol. 16, no. 2, pp. 279–286, Mar. 2012.

[3] D. He, C. Chen, S. Chan, and J. Bu, "DiCode: DoS-resistant and distributed code dissemination in wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 11, no. 5, pp. 1946–1956, May 2012.

[4] V. Shnayder, B.-R. Chen, K. Lorincz, T. R. F. Fulford-Jones, and M. Welsh, "Sensor networks for medical care," Harvard Univ., Cambridge, MA, USA, Tech. Rep. TR-08-05, 2005

[5] Crossbow Solutions Newsletter. Motes for mobile communication and tele-medicine. 2005.

[6] K. Malasri and L. Wang, "Addressing security in medical sensor networks," in *Proc. ACM HealthNet*, 2007, pp. 7–12.

[7] R. Rajasekaran, V. Manjula, V. Kishore, and T. Sridhar, C. Jayakumar, "An efficient and secure key agreement scheme using physiological signals in body area networks," in *Proc. Int. Conf. Advances Comput. Commun. Informat.*, 2012, pp. 1143–1147.

[8] H. Wang, H. Fang, L. Xing, and M. Chen, "An integrated biometric-based security framework using wavelet-domain HMM in wireless body area networks (WBAN)," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2011, pp. 1–5.

[9] K. Malasri and L. Wang, "Design and implementation of a secure wireless mote-based medical sensor network," *Sensors*, vol. 9, no. 8, pp. 6273–6297, Aug. 2009.

[10] S. Keoh, "Efficient group key management and authentication for body sensor networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2011, pp. 1–6.

[11] C. C. Tan, H. Wang, S. Zhong, and Q. Li, "IBE-lite: A lightweight identity-based cryptography for body sensor networks," *IEEE Trans. Inf. Technol. Biomed.*, vol. 13, no. 6, pp. 926–932, Nov. 2009.

[12] X. Le, M. Khalid, R. Sankar, and S. Lee, "An efficient mutual authentication and access control scheme for wireless sensor network in healthcare," *J. Networks*, vol. 6, no. 3, pp. 355–364, 2011.

[13] P. Kumar and H.-J. Lee, "Security issues in healthcare applications using wireless medical sensor networks: A survey," *sensor*, vol. 12, pp. 55–91, 2012.

[14] D. He, J. Bu, S. Zhu, S. Chan, and C. Chen, "Distributed access control with privacy support in wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 10, no. 10, pp. 3472–3481, Oct. 2011.

[15] W. He, Y. Huang, R. Sathyam, K. Nahrstedt, and W. Lee, "SMOCK: A scalable method of cryptographic key management for mission-critical wireless ad-hoc networks," *IEEE Trans. Inf. Forensics Security*, vol. 4, no. 1, pp. 140–150, Mar. 2009.

[16] S. Zhao, A. Aggarwal, R. Frost, and X. Bai, "A survey of applications of identity-based cryptography in mobile ad-hoc networks," *IEEE Commun. Surveys Tutorials*, vol. 14, no. 2, pp. 380–400, Second Quarter 2012.

[17] D. Ma and G. Tsudik, "Security and privacy in emerging wireless networks," *IEEE Wireless Commun.*, vol. 17, no. 5, pp. 12–21, Oct. 2010.

[18] C. Cordeiro and M. Patel, "Body area network standardization: Present and future directions," in *Proc. BodyNets*, 2007, pp. 1–2.

[19] Z. Shao, "Provably secure proxy-protected signature schemes based on RSA," *Comput. Electr. Eng.*, vol. 35, no. 3, pp. 497–505, May 2009.

[20] Z. Shao, "Proxy signature schemes based on factoring," *Inf. Process. Lett.*, vol. 85, no. 3, pp. 137–143, 2003.

[21] K. C. Barr and K. Asanovi, "Energy aware lossless data compression," *ACM Trans. Comput. Syst.*, vol. 24, no. 3, pp. 250–291, Aug. 2006.

[22] OpenSSL, [Online]. Available: http://www.openssl.org

[23] TinyOS: An open-source OS for the networked sensor regime. [Online]. Available: http://www.tinyos.net/

[24] J. Lee, K. Kapitanova, and S. Son, "The price of security in wireless sensor networks," *Comput. Networks*, vol. 54, no. 17, pp. 2967–2978, Dec. 2010.

[25] A. Liu and P. Ning, "TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks," in *Proc. Inf. Process. Sensor Netw.*, 2008, pp. 245–256.

[26] Software AES, [Online]. Available: http://tinyos.cvs.sourceforge.net/viewvc/tinyos/tinyos-2.x-contrib/crypto/index.html

[27] A. Milenkovi, C. Otto, and E. Jovanov, "Wireless sensor networks for personal health monitoring: Issues and an implementation," *Comput. Commun.*, vol. 29, no. 13–14, pp. 2521–2533, Aug. 2006.

Daojing He received the B.Eng. and M. Eng. degrees from the Harbin Institute of Technology, Harbin, China in 2007 and 2009, respectively, and the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2012, all in Computer Science.

He is currently an Associate Professor at the South China University of Technology, Guangzhou, China. His research interests include network and systems security.

Dr. He is an Associate Editor or on the editorial board of some international journals such as *Springer Journal of Wireless Networks*, *Wiley's Wireless Communications and Mobile Computing Journal*, *Wiley's Security and Communication Networks Journal*, and *KSII Transactions on Internet and Information Systems*. He has been serving as a TPC member for leading conferences including IEEE Wireless Communications and Networking Conference, the IEEE Global Communications Conference, and the IEEE International Conference on Communications.

Sammy Chan (S'87–M'89) received the B.E. and M.Eng.Sc. degrees in electrical engineering from the University of Melbourne, Parkville, Vic., Australia, in 1988 and 1990, respectively, and the Ph.D. degree in communication engineering from the Royal Melbourne Institute of Technology, Melbourne, Vic., Australia, in 1995.

From 1989 to 1994, he was with Telecom Australia Research Laboratories, first as a Research Engineer, and between 1992 and 1994 as a Senior Research Engineer and a Project Leader. Since December 1994, he has been with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, where he is currently an Associate Professor.

Shaohua Tang (M'98) received the B.Sc. and M.Sc. degrees in applied mathematics in 1991 and 1994, respectively, and the Ph.D. degree in communication and information system from South China University of Technology, in 1998, all from the South China University of Technology, Guangzhou, China.

He was a Visiting Scholar with North Carolina State University, Raleigh, NC, USA, during 2001–2002, and a Visiting Professor with the University of Cincinnati, Cincinnati, OH, USA, during 2009–2010. He has been a Full Professor with the School of Computer Science and Engineering, South China University of Technology since 2004. His current research interests include information security, networking, and information processing.

Dr. Tang is a member of the IEEE Computer Society.