Design and Implementation of a Secure Wireless Mote-Based Medical Sensor Network [*†]

Kriangsiri Malasri, Lan Wang
Department of Computer Science
University of Memphis
Memphis, TN 38152-3240
{kmalasri, lanwang}@memphis.edu

ABSTRACT

A *medical sensor network* can wirelessly monitor vital signs of humans, making it useful for long-term health care without sacrificing patient comfort and mobility. For such a network to be viable, its design must protect data privacy and authenticity given that medical data are highly sensitive. We identify the unique security challenges facing such a sensor network and propose a set of resource-efficient mechanisms to address these challenges. Our solution includes (1) a novel two-tier scheme for verifying the authenticity of patient data; (2) an ECC-based secure key exchange protocol to set up shared keys between sensor nodes and base stations; and (3) symmetric encryption/decryption for protecting data confidentiality and integrity. We have implemented the proposed mechanisms on a wireless mote platform and our results confirm their feasibility.

ACM Classification Keywords

C.2.0 Computer-Communication Networks: General—*security and protection*; J.3 Computer Applications: Life and Medical Sciences—*medical information systems*

General Terms

Design, Experimentation, Security, Human Factors

Author Keywords

health monitoring, privacy, authenticity, sensor network

INTRODUCTION

In our society, an increasing percentage of the population are aging people who have chronic illnesses such as diabetes and heart disease. In addition, more and more children are suffering from long-term conditions like asthma and obesity. If these people's health could be monitored continuously over a long period of time, physicians could provide more accurate diagnoses and better treatment. For instance, studies (e.g., [1, 2]) have shown that monitoring patient data can help with early detection of conditions like heart disease. Moreover, medical professionals could react to situations such as strokes and asthma attacks more quickly. Current monitoring solutions, however, are not suitable for long-term use, as patients are typically attached to a bedside device that limits their mobility and comfort.

Several research groups (e.g., [3, 4, 5, 6, 7]) have recently integrated medical sensors with wireless *motes* for health monitoring, such as the Harvard wireless pulse oximeter [4]. A mote is low-power computing device with a wireless radio; it is typically the size of a match box or even smaller, such as the Crossbow MICA motes [8]. Using the motes, these medical sensors wirelessly transmit data to base stations, where it can be accessed by physicians and nurses. This frees patients from the confinement of traditional wired sensors, allowing medical professionals to monitor their health remotely over long periods. Such *medical sensor networks* can be deployed in hospitals, long-term care facilities, and homes. Pharmaceutical companies could also use the network to monitor patients in clinical trials in order to develop better drugs. Moreover, the system can be extended to monitor the vital signs of people working in hazardous conditions, such as firefighters in a burning building, relief workers in a disaster area, and soldiers on a battlefield.

Although medical sensor networks are extremely useful and versatile, the medical data they collect is sensitive, and the privacy of such data is protected by laws (e.g., the Health Insurance Portability and Accountability Act of 1996 (HIPAA) [9]). A patient's vital signs may reveal what disease the patient has (which might be useful to parties such as insurance companies). As such, an attacker may profit financially by selling data obtained through eavesdropping. Moreover, the attacker could even cause physical harm to a patient by misreporting or spoofing the patient's data, resulting in improper diagnosis and/or treatment. Therefore, it would be irresponsible to design and deploy a medical sensor network without adequate security mechanisms. To this end, our work aims to design a secure medical sensor network for health monitoring. We believe that security mechanisms must be designed

[*] A preliminary version of this work appeared in the 1st International Workshop on Systems and Networking Support for Healthcare and Assisted Living Environments (HealthNet), June 2007.

[†] This work was supported in full or in part by a grant from The University of Memphis Faculty Research Grant Fund. This support does not necessarily imply endorsement by the University of research conclusions.

UbiComp'08, September 21-24, 2008, Seoul, Korea.

into the architecture from day one rather than after the other issues are addressed, as security requires careful thought on where functionality should be placed and how the system components interact with one another.

Similar security issues may exist in some traditional wireless ad hoc networks and other types of sensor networks. However, a medical sensor network monitors *humans*, unlike most existing sensor networks that monitor the *physical environment*. A *human-centered* sensor network has distinct features such as the sensitive nature of the data, the mobility of sensors, and the proximity to potential attackers, which make it difficult to address security.

The contribution of our work is the following. First, we identify the security requirements and challenges in a medical sensor network. Second, we propose an architecture for wireless motes-based health monitoring and the following security mechanisms: (1) a two-tier authentication scheme to ensure the authenticity of patient data; (2) a secure key exchange protocol to set up symmetric keys between sensor nodes and base stations; and (3) symmetric encryption and decryption for protecting data confidentiality and integrity. Third, we have developed a prototype on the Tmote Sky platform for evaluating the security, cost, and performance of the proposed architecture and mechanisms. Note that our design could be applied in a variety of scenarios, e.g., hospitals, assisted living facilities, and homes. However, we use the terminology *patient*, *physician*, and *healthcare facility* for simplicity.

DESIGN REQUIREMENTS

Our overall objective is to build a *secure* medical sensor network for health monitoring using *low-power* wireless motes. To achieve this objective, the system needs to support long-term monitoring while ensuring the privacy and authenticity of medical data.

First, patients with chronic diseases require **long-term monitoring**, which implies the following: (a) the sensor nodes should ideally operate for days without recharging; and (b) during the lifetime of a sensor node, it may be turned off and on for various reasons, e.g. when it is out of battery power, so the design should not rely on the assumption that the sensor nodes are always on. These considerations have major influences on the design of our security mechanisms.

Second, since personal health data is extremely sensitive, a high confidence on the system's ability to ensure **data privacy** is essential. From healthcare providers' point of view, a data breach would have very serious legal and financial consequences. From users' point of view, they may be hesitant to use the system if they are not convinced that their data will be kept confidential.

Third, **data authenticity** is very important in our system since healthcare providers rely on the data to diagnose and treat their patients as well as react to emergencies. Patients' lives can be endangered if their data or commands from healthcare providers are intentionally changed or forged. Note that verification of data authenticity has to be carried out as soon as data is received to ensure timely reaction to emergencies.

We emphasize that our work focuses on keeping patient data secure *as it is transferred to base stations*. The problem of data security does not stop at that point; there are still issues such as ensuring that only authorized personnel can view the data, and preventing accidental disclosure of the data. These areas are outside the scope of our work. We do note, however, that there has been work addressing these issues, e.g. several variants of role-based access control (RBAC) tailored for the healthcare industry surveyed in [10].

UNIQUE CHALLENGES

We have chosen to use wireless motes (such as the Crossbow MICA [8] and the Moteiv Tmote Sky [11]) as the sensor nodes in our design due to two main reasons. First, the motes are much smaller and lighter than other types of portable embedded devices (e.g. PDAs), making them easier to carry. Second, motes consume much less energy than other devices, so they are better candidates for long-term monitoring. The Tmote Sky, for instance, draws only 22 mA of current while transmitting or receiving, and a mere 1.8 mA to operate the onboard processor without the wireless radio [11].

Unfortunately, *the resource constraints in wireless motes make it challenging to meet the above-mentioned security requirements*. Motes have much lower processor speed, memory, link bandwidth, and energy supply than mobile PCs or PDAs, so our security mechanisms must be very resource-efficient. The MICA motes, for example, use an 8-bit, 8 MHz processor, and the comparable Tmote Sky platform employs a 16-bit, 8 MHz processor. Conventional security mechanisms incur high costs; some RSA operations can take over 80 seconds to run on a low-power 8-bit CPU [12]. Although the recent Intel iMote platform has higher processor speed and more RAM, it also consumes more energy than the previously mentioned motes. In addition, the radio capabilities of the latest IEEE 802.15.4-compliant motes are limited to bandwidths of 250 kbps.

Furthermore, security mechanisms developed for other sensor networks or in general may not be applicable to medical sensor networks due to their unique characteristics. First, *sensor nodes on different patients do not communicate with one another*. Instead, they communicate with one or more base stations close to the healthcare provider. Therefore, we need to protect the communication between the sensor nodes and base stations. However, most existing research focuses more on enabling secure communication among sensor nodes. Often, a very simple symmetric-key mechanism is used between the sensor nodes and base stations, e.g. preconfiguring a shared secret in each sensor node and the base station. These mechanisms do not offer enough protection against physical compromise of sensor nodes. Second, *physical compromise of sensor nodes* is a greater threat in medical sensor networks due to two factors: (1) *mobility*: while most other sensor networks have stationary sensors, medical sensors move with the patients and may get lost without even being noticed; (2) *accessibility*: it is relatively easy for

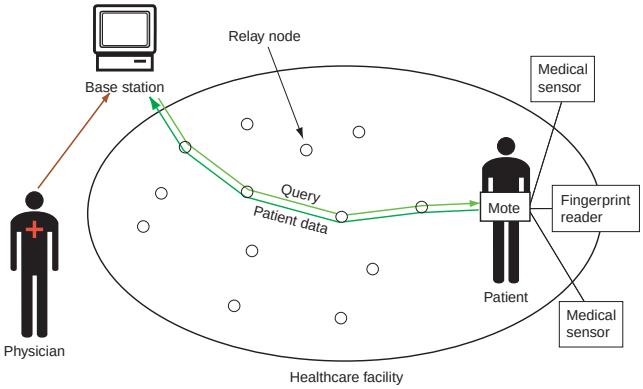

Figure 1. Wireless health monitoring architecture.

attackers to find targets, i.e. patients carrying medical sensors, in local healthcare facilities, as opposed to in remote locations such as forests or battlefields. Note that physically compromising the base stations is harder because they can be protected in secure locations.

Once a sensor node is compromised, the attacker can capture any sensitive information remaining in the mote, including personal medical data and secret key material. The attacker can then use the captured key material to decrypt any previously recorded communication. Furthermore, the attacker can inject false data using the node. Therefore, we need to limit the amount of private information disclosed after the physical compromise as well as the damage that the attacker can cause by impersonating the patient. This has several implications for our design: (1) secret keys should be periodically updated; (2) when a node is turned off, no secret key material should have to be stored permanently in non-volatile memory in the sensor nodes (a pre-configured shared secret obviously does not satisfy this requirement); and (3) authentication must rely on something that the attacker cannot easily capture or forge.

DESIGN OVERVIEW

Fig. 1 illustrates our overall architecture. Each *patient* has one wireless *mote* attached to his/her body. The mote is connected (wired) to several *medical sensors*, which take samples of the patient's health data when they are activated. A *medical professional* issues queries for a patient's data through one or a few *base stations*. Queries issued from base stations may activate the patient's medical sensors or adjust their sampling frequency and other parameters. Before sending the queries, the base station verifies that the medical professional is a legitimate user with the necessary privileges to access the particular patient's data. After the mote receives the query from the base station, it activates the appropriate sensor(s) or adjusts its parameters. It also continuously sends the resulting patient data from the sensors to the base station until the sensors are deactivated. Wireless *relay nodes* forward the queries and patient data between the base stations and the motes.

Architectural Decisions

We have made two explicit architectural decisions that differentiate our design from previously proposed architectures

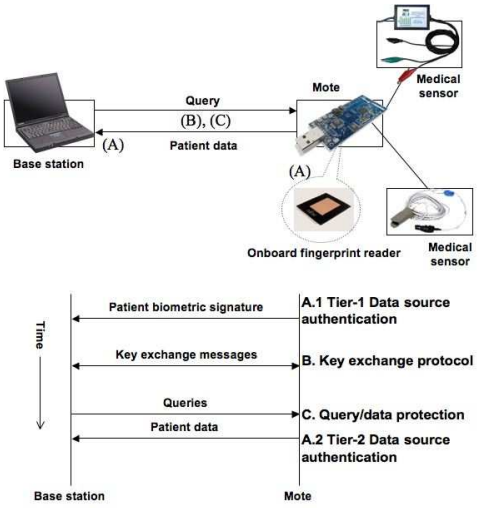

Figure 2. Three security mechanisms.

such as CodeBlue [4]. These choices make it easier to secure the overall system, without unduly sacrificing the system's functionality or performance.

First, *the motes communicate only with the base stations, not with individual medical professionals' computers*. This is because in a large healthcare facility, it would be difficult for the motes to directly authenticate hundreds of individual users (physicians, nurses, etc.). The motes simply do not have the memory resources to maintain all this access control information. Instead, we decide to authenticate the users at the base stations and have the base stations issue queries to the motes on behalf of the users. Motes are configured with the base stations' addresses and public keys for authentication.

Second, *motes do not have their own permanent public/private key pairs*, because such a key pair would have to be kept in non-volatile memory to prevent it from being lost when a mote is turned off. Since individual motes are relatively easy to physically compromise, an attacker who captures a mote would be able to decrypt all previously recorded data sent by the mote. Moreover, if the mote's public/private key pair is used for authentication, the attacker would be able to impersonate the patient after capturing the mote.

Security Mechanisms

Our security mechanisms are summarized in Fig. 2. First, we propose a *two-tier authentication scheme* based on patient biometric and physiological data, in order to handle spoofing or physical compromise of motes. We assume that each mote is connected to a small fingerprint scanner or a comparable biometric identification device. The base station first verifies the patient's identity via a biometric signature, then checks incoming data against the patient's profile for consistency. Our authentication scheme is resilient against spoofing and physical compromise of motes, because it is difficult for an attacker to both forge a valid patient's biometric signature and generate physiological data that is consistent with the patient's own data.

Second, *queries and data are encrypted and integrity protected using dynamically generated symmetric keys*, in order to defend against attacks that compromise the privacy and integrity of patient data. As mentioned earlier, motes are very resource-constrained, so it is more efficient to use pairwise *symmetric keys* shared between the base station and motes for data protection.

Third, we propose an *Elliptic Curve Cryptography (ECC) based key exchange protocol* to allow base stations and motes to securely derive symmetric keys without any prior shared secrets. In order to handle spoofing of base stations, motes are pre-configured with base stations' public keys so that they can establish symmetric keys only with valid base stations[1]. In the next three sections, we present the specific design, implementation, and evaluation of each mechanism.

TWO-TIER DATA SOURCE AUTHENTICATION SCHEME

An attacker can inject forged patient data into a medical sensor network using either his/her own motes or a compromised mote. To combat this type of attack, we propose a two-tier data source authentication scheme.

Design

At the first tier, each mote is integrated with a small biometric scanner to capture a unique signature for the patient. The biometric signature must be validated by a base station before communication between the base station and mote can occur. We assume that the base stations have access to a list of valid patient biometric signatures (e.g., this could be obtained from each patient upon admission to the facility), and also that the space of possible signatures is sufficiently large that a brute-force attack is infeasible. There are several off-the-shelf miniature fingerprint readers and finger vein readers that can serve as the biometric scanner. One example is the MBF200 solid-state fingerprint sensor from Fujitsu [14] (also shown in Fig. 2). It is small (24 mm $\times$ 24 mm $\times$ 1.4 mm), low in power consumption (20 mA active, $<$ 200 μA sleep, 20 μA standby), and inexpensive (about $30 per unit in bulk quantities). Moreover, it can capture a fingerprint in less than one second. The fingerprint sensor alone simply captures an image (which would be inefficient to transmit in its raw form), but off-the-shelf hardware (e.g., [15]) exists that integrates the Fujitsu sensor with a small processing board to allow feature extraction from the raw image.

The first tier alone is insufficient for ensuring data authenticity because an attacker can capture a patient's mote after the patient has been accepted into the system. Therefore, we add a second-tier authentication system to *assert the identity of a patient based solely on the sensor data being collected from that patient*. More specifically, each patient will wear the sensors for a short period of time for the base station to learn the patient's profile. From then on, whenever the patient's data deviates from that profile, the base station will raise an alarm. The alarm could be caused by forged data or by a medical emergency. In either case, it is important that the patient be checked on, so issuing the alarm is warranted. This approach does not require any extra bandwidth (the data being used is sent anyway) and is completely passive from the viewpoint of the patient.

[1] If there are many base stations in the facility, we can use the *group public-key scheme* proposed in [13]. This scheme allows each base station to have its own private key, while allowing the motes to use a *single* public key for the entire base station group.

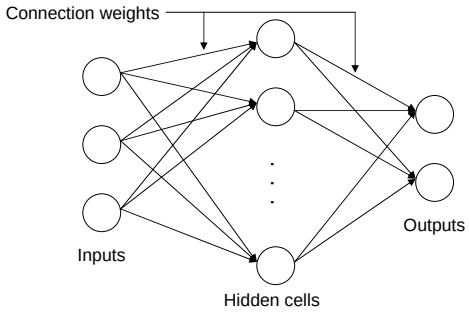

Figure 3. A generic three-layer feed-forward neural network.

Implementation of Tier-2 Authentication

Our second-tier authentication mechanism requires statistical or machine learning techniques to recognize medical data as being from a particular patient. We have investigated a neural network approach for implementing this mechanism. Each patient is associated with a neural network, which is first trained on data from that patient and some reference patients. The neural network then monitors incoming data for consistency with the training data. Both the initial training and detection mechanism are run on the base station, to minimize computation on the mote. Once trained, the neural network can quickly process incoming data.

We use a feed-forward neural network (Fig. 3) that consists of a number of inputs, hidden cells, and outputs, along with a set of *connection weights* that determine how the output is computed from the input. Based on known input-output pairs (called the *training set*), the weights can be adjusted to cause the neural network to produce the desired output when presented with the associated input. This process is called *training*; the procedure used to adjust the weights is the *learning algorithm*. Ideally the trained network should be capable of some generalization: it should give a reasonable output even for input outside of its training set.

Our current implementation uses patient electrocardiogram (ECG) signals to train the neural network, due to the many identifying features that this data offers. The input to the neural network is one heartbeat of ECG data; the output is 0 or 1 (NO or YES) depending on whether the network believes this data is consistent with previously seen data.

One issue with this approach is that raw ECG data for a single heartbeat can contain hundreds of individual samples. Using this data in its entirety is impractically complex. To address this problem we opted to extract a set of representative features from the raw ECG data to use as neural network inputs. A typical ECG signal for a single heartbeat is shown in Fig. 4; it consists of a small *P wave*, followed by a large *QRS complex*, followed by a small *T wave*. We extract eight features per heartbeat: the lengths of the (1) PR interval, (2)

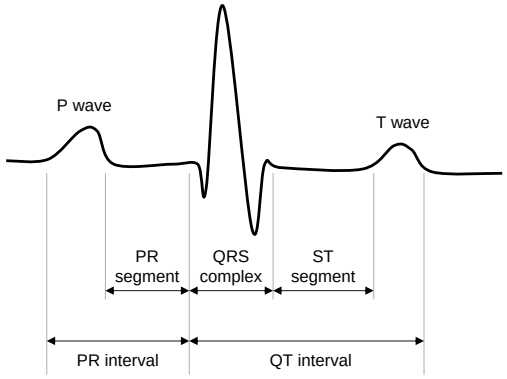

Figure 4. A typical ECG signal for a single heartbeat.

Structure	Features	Ref. Patients	Accuracy
8×20×1	8	5	69.79%
8×20×1	8	9	81.86%
14×20×1	14	5	80.07%
14×20×1	14	9	84.44%
8×15×15×1	8	9	91.31%
14×15×15×1	14	9	89.66%

Table 1. Comparison of Neural Network Setups
Structure refers to the number of input, hidden, and output cells in the neural network. Ref. Patients refers to the number of reference patients. The accuracies represent the average validation accuracy over *all* patients.

PR segment, (3) QRS complex, (4) ST segment, and (5) QT interval; and the peak amplitudes of the (6) P wave, (7) QRS complex, and (8) T wave. In some experiments, we used a total of 14 features; this includes 6 additional amplitudes of the ECG signal (selected at roughly the midpoints of the begin-peak and peak-end segments of the P wave, QRS complex, and T wave) as well as the 8 features above.

Evaluation of Tier-2 Authentication

As we do not yet have access to real ECG sensors, we used publicly available data from the long-term ST ECG database at MIT's PhysioWeb [16] for developing our neural networks. We selected 19 patients' records, each containing ECG data for that patient over a period of 21-24 hours. Some of these patients (denoted $R_1 ... R_i$) were set aside as a "reference set." For each of the remaining patients (denoted $P_1 ... P_j$), a training data set was constructed consisting of the first 2000 heartbeats from that particular patient's record (associated with a neural network output of 1) and the first 2000 heartbeats from each of the patients in the reference set (associated with a neural network output of 0).

We performed some experimentation with different neural network setups by varying the number of reference patients used in training as well as the features extracted from each beat of ECG data. The results in Table 1 suggest that increasing the number of reference patients potentially has the greatest returns on performance. For subsequent experiments, we chose the setup with the best overall performance for the training of each patient's neural network, i.e. 9 reference patients and a four-layer feed-forward neural network model with 8 input cells, 2 layers of 15 hidden cells, 1 output cell, and a log sigmoid activation function on the hidden layers.

Table 2 shows how well the trained neural networks could classify the training data. The number corresponding to $(P_i$, self or $R_j)$ indicates the percentage of training samples (heartbeats) from self or R_j that were correctly classified; samples from self should be classified as from patient P_i, while samples from R_j should not. For example, the sixth row of Table 2 indicates that the trained network for patient P_6 was able to correctly classify 99.45% of the training inputs from P_6 as being from P_6 (the first value); and it was able to classify 100% of the training inputs from the reference patients R_1, R_2, R_4, R_5, R_6, R_7, and R_9, and 99.95% of the training inputs from R_3 and R_8, as *not* being from P_6. Overall, the neural networks learned the training data very well.

Next, we use each trained neural network to classify the validation data (i.e., 10,000 heartbeats not used in training) from each of the non-reference patients. As shown in Table 3, the neural networks give accurate results in most cases. For example, the sixth row shows that patient P_6's neural network was able to correctly classify the validation data from patients P_1, P_3 through P_8, and P_{10} almost 100% of the time, but was less accurate with P_2 and P_9.

It is important to note that perfect classification is *not* required at all in our scheme since our goal is to detect suspicious data. For example, suppose that an alarm is raised if at least 20% of incoming data deviates from a patient's profile. Such a low threshold would not cause us to suspect a valid patient. As shown in Table 3, the lowest value of (P_i, P_i), $i = 1, ..., 10$, is 86.62%, which means at most 13.38% of a patient's data is misclassified as not from the patient. Therefore, a 20% threshold would not incorrectly trigger an alarm. At the same time, all of the (P_i, P_j), $i \neq j$, are greater than 20%. This means that patient P_i's neural network can identify at least 20% of P_j's data as not from P_i, correctly raising an alarm. Of course, this example is limited to only the patient data that we considered, but the same reasoning could be applied to a larger sample.

ECC-BASED KEY EXCHANGE PROTOCOL

We propose a key exchange protocol that each mote uses to securely derive symmetric keys with the base station at the beginning of their communication. We use *elliptic curve cryptography* (ECC) [17, 18, 19] in our key exchange protocol rather than the well-known RSA, as it offers comparable security for a much smaller key length (a 160-bit ECC key is roughly equivalent to a 1024-bit RSA key). Moreover, the computational efficiency of ECC operations is comparable to or better than that of their RSA counterparts [12].

Design

We first present some background on ECC and then describe the design of our key exchange and key update protocols.

ECC Background

An *elliptic curve* is of the form $y^2 = x^3 + ax + b$. When defined over a finite field, all the points on the curve (x, y) and the parameters a and b are limited to elements of the underlying field. A common class of finite fields used in ECC are *prime fields* $GF(p)$, where p is a large prime number;

Patient	Self	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9
P_1	99.95	100	100	100	100	100	100	100	100	100
P_2	99.95	100	100	100	100	100	100	100	100	100
P_3	98.75	100	100	99.55	99.95	99.85	100	100	99.85	100
P_4	99.75	100	99.95	100	100	99.65	100	100	100	99.95
P_5	98.80	100	100	100	100	100	100	100	99.85	99.70
P_6	99.45	100	100	99.95	100	100	100	100	99.95	100
P_7	98.80	100	100	100	100	99.60	99.90	100	100	99.95
P_8	100	100	100	100	100	100	100	100	100	100
P_9	100	100	100	100	100	100	100	100	100	100
P_{10}	99.05	100	100	100	100	99.90	99.95	100	99.50	100

Table 2. Neural Network Results on Training Data

Patient	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
P_1	94.87	99.66	81.76	93.52	69.04	99.52	99.72	30.11	99.61	99.19
P_2	99.98	99.16	100	99.46	99.98	64.51	99.29	100	53.50	96.84
P_3	85.80	99.23	89.52	77.65	92.16	98.35	99.86	58.93	99.99	99.54
P_4	91.13	87.07	77.49	96.23	88.31	80.88	99.52	91.23	99.35	99.37
P_5	73.22	99.96	99.04	94.26	91.24	99.93	98.62	96.92	99.82	99.54
P_6	98.54	48.87	94.21	97.58	99.04	95.45	99.71	99.53	52.99	98.75
P_7	99.96	96.50	99.49	99.36	97.46	96.54	87.63	99.77	88.53	98.66
P_8	99.74	100	99.98	100	97.67	100	99.97	99.87	100	99.61
P_9	99.92	79.97	99.89	98.07	99.76	83.62	81.42	100	97.40	98.33
P_{10}	99.89	36.78	99.87	97.95	98.57	61.85	98.77	100	23.15	86.62

Table 3. Neural Network Results on Validation Data

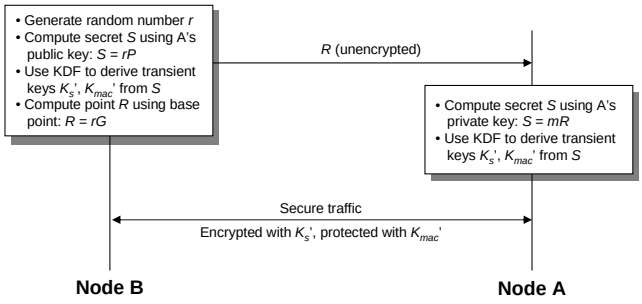

Figure 5. The Elliptic Curve Integrated Encryption Scheme (ECIES).

the elements of this field are the integers in $[0, p-1]$. There are three basic operations that can be performed on elliptic curve points: *addition* of two points $P+Q$, *doubling* of a point (addition to itself) $2P$, and *scalar multiplication* nP, where n is an integer. Algebraic formulas exist for both addition and doubling. Scalar multiplication is the repeated application of addition and doubling: $nP = P+P+...+P$ for n times. To use ECC, two nodes A and B agree on what elliptic curve to use and a base point G on the curve; this information is not secret. Node A can generate a large random number m as its private key and derive its public key P by using $P = mG$. If m is large, it is hard for an attacker to derive the private key m from the public key P even if the attacker knows G, due to the difficulty of the *elliptic curve discrete logarithm problem* (ECDLP).

In our scheme, a mote uses the base station's public key to secure the *key exchange protocol messages* between the base station and itself, using the *Elliptic Curve Integrated Encryption Scheme* (ECIES) [19]. The ECIES procedure is summarized in Fig. 5 and described below in more detail. To protect the messages from node B (a mote) to node A (base station) against eavesdropping and modification, node B generates a random number r and computes a secret $S = rP$. Node B then uses a key derivation function (KDF) to generate two symmetric *transient keys* from S: an encryption key $K_s^{'}$ and a MAC (message authentication code) key $K_{mac}^{'}$. Node B also computes the point $R = rG$, which is sent in the clear to node A. Node A can derive the secret S using its private key m and the point R: $S = mR = m(rG) = r(mG) = rP$. Node A then uses the same KDF to derive $K_s^{'}$ and $K_{mac}^{'}$ from S. The two transient keys are then used to encrypt and integrity-protect the messages between B and A. Again, due to the ECDLP's difficulty, it is infeasible for an eavesdropper to derive r (and hence S) even with knowledge of R and G, if r is sufficiently large.

Key Exchange Protocol

In our scheme, each base station has a private key m_{BS} and a public key $P_{BS} = m_{BS}G$, where G is a chosen base point on the elliptic curve. The public key P_{BS} is pre-configured in the motes. As mentioned before, motes do not have their own public/private keys. To securely derive symmetric keys between a mote and a base station (called *session keys*), we use the following protocol involving three messages (Fig. 6).

When a mote is attached to a patient, the patient uses the fingerprint scanner on the mote to activate the key exchange

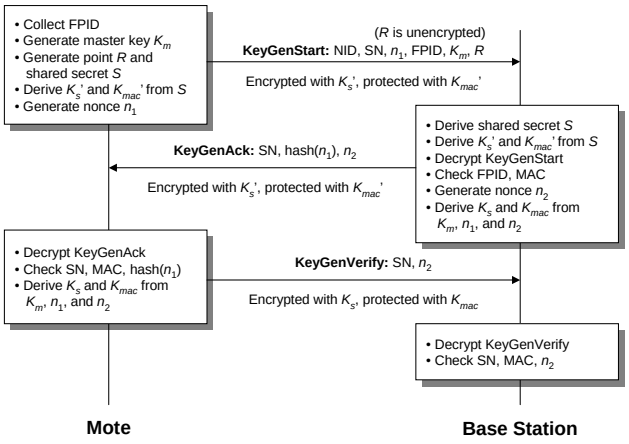

Figure 6. ECC-based key exchange protocol.

protocol. For illustration, we assume that a fingerprint reader is used to obtain a biometric signature, but other types of biometric information would also work. Once the patient's fingerprint ID ($FPID$) is obtained, the mote generates a master key K_m that will be used to derive the session keys for end-to-end query/data encryption. It also generates a session number SN that uniquely identifies this particular communication session, and a nonce n_1 for deriving the session keys. The mote then sends a *KeyGenStart* message *securely* to the base station using ECIES. This message contains the mote's ID (NID), the patient's $FPID$, the master key K_m, the session number SN, and the nonce n_1. Appended to *KeyGenStart* is the unencrypted point R needed for ECIES.

The base station decrypts *KeyGenStart* and verifies the authenticity of this patient using the fingerprint ID. Then the base station generates a nonce n_2, and it uses K_m, n_1, and n_2 to derive the session keys that will be shared with the mote – the symmetric encryption key K_s and the MAC key K_{mac} for data/query transfer. The base station then sends back a *KeyGenAck* message to the mote containing SN, n_2, and a one-way hash of n_1, encrypted and integrity-protected using the transient keys derived from ECIES[2]. The mote decrypts *KeyGenAck* and verifies SN and the hash of n_1 to prevent replay attacks. Afterwards, the mote uses K_m, n_1, and n_2 to derive the session keys K_s and K_{mac}, which should match those generated by the base station.

For the base station to verify that the mote has received the *KeyGenAck* message and that the mote generated the session keys K_s and K_{mac} correctly, the mote sends a *KeyGenVerify* message to the base station containing SN and n_2. This message is encrypted with K_s and integrity-protected with K_{mac}. The base station decrypts *KeyGenVerify* and verifies that all the values are correct. If so, data transfer may commence using K_s to encrypt/decrypt messages and, if desired, K_{mac} to compute a keyed MAC for each message.

Our key exchange protocol shares some similarities with handshakes in SSL. The major differences are that (1) we use only three protocol messages to set up the session keys, as message transmissions consume a significant amount of energy; (2) since motes do not have their own public/private keys, we incorporate the patient biometric signature in *KeyGenStart* as a means of authenticating the mote to the base station; and (3) in addition to the patient biometric signature, we have several other built-in measures to limit what an attacker can do with a compromised mote.

Key Update Procedure

The base station and the mote periodically update the session keys K_s and K_{mac} to limit the amount of private data that can be recovered in case the keys are compromised. To update the session keys, they simply exchange new values of n_1 and n_2 and rerun the key derivation function using the existing master key K_m and the new nonce values. This allows the session keys to be updated without having to undertake the public-key operations in the full key exchange protocol. We omit the details due to space constraints.

Further Considerations for Physical Compromise of Motes

We enhance the above mechanisms with the following measures to limit what an attacker can do with a compromised mote. Suppose that an attacker has been eavesdropping on the communication between the mote and the base station. We have two measures to bound the amount of private information that the attacker can recover from earlier captured sensor data. First, once the key update procedure produces a new session key, the previous session's keys are erased from memory. Therefore, the attacker can only decrypt the data that has already been sent in the *current* session. Second, if the base station and the mote have not communicated for a long period of time (this value could be tuned as necessary), the master key K_m and its derived keys expire and are removed from memory. The attacker will not obtain any session keys from the mote in this case.

Suppose that the attacker wants to send forged data using the mote. If the mote has been idle for too long and the master key has expired, the attacker needs to use a valid fingerprint ID to activate the key exchange protocol. To prevent this from happening, we remove $FPID$ from the mote's memory as soon as the *KeyGenAck* message is received, so it is difficult for an attacker to obtain a valid patient's fingerprint information. On the other hand, if the master key is still valid, the attacker can use the current session key and establish new session keys through the key update procedure. However, s/he will most likely be detected by the base station via the second-tier authentication scheme, as the forged patient data will not match the existing profile.

Implementation of Key Exchange Protocol

Our development environment is TinyOS 2 [20] running on the Moteiv Tmote Sky platform[3], which features a 16-bit, 8

[2]We opt to use a hash of n_1 rather than n_1 directly in order to minimize the chance of an eavesdropper with a compromised transient key deriving the correct session keys.

[3]As of late 2007, Moteiv has changed its name to Sentilla and has discontinued production and support of its Tmote product line in favor of a new hardware platform designed for Java applications. However, the new platform is backwards-compatible with the Tmote Sky. Furthermore, Crossbow still offers its TelosB mote for sale, which is functionally identical to the Tmote Sky.

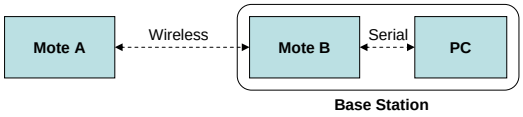

Figure 7. Experimental setup with two motes and a PC.

MHz Texas Instruments MSP430 processor with 48 KB of program ROM and 10 KB of RAM. Program code in this environment is written in nesC (networked embedded systems C), a dialect of C designed for use with TinyOS.

Our basic ECC operations are partly based on NCSU's TinyECC code for the Crossbow MICA [21]. We modified TinyECC to run on the Tmote Sky by replacing the inline assembly with MSP430 assembly, making use of the MSP430's hardware multiplier, and incorporating a fast modular inversion algorithm involving only bit shifts and additions [22].

We implemented ECIES for secure key exchange, using the elliptic curve secp160r1 defined over a 160-bit prime field as recommended by [23]. We use 160-bit private keys, 320-bit public keys, and a 160-bit random number r in ECIES. We then implemented our key exchange protocol, using a 32-bit session number (SN), 32-bit patient ID ($FPID$), 128-bit master key (K_m), 128-bit transient and session keys ($K_s^{'}$, $K_{mac}^{'}$, K_s, K_{mac}), and 64-bit nonces (n_1, n_2). The resulting *KeyGenStart*, *KeyGenAck*, and $KeyGenVerify$ messages are 54, 52, and 32 bytes long.

Evaluation of Key Exchange Protocol

Our entire code on the Tmote uses 24.7 KB of ROM and 2.8 KB of RAM, which leaves ample room for other applications. The ECC code is fairly efficient – it carries out a 160-bit scalar point multiplication (the most computationally intensive operation in ECC), in 5.3 s, which is almost 15% faster than TinyECC's original result. However, we note that TinyECC has since undergone some revisions that have improved its performance.

We timed the protocol running on two motes and a laptop PC (2.66 GHz Intel Pentium 4) acting as a base station. As illustrated in Fig. 7, mote A acts as a sender, while mote B serves as the interface between the sender and the PC by receiving A's wireless packets and forwarding them over a serial connection (emulated via a physical USB connection). Mote A took 11.4 s to generate and send *KeyGenStart*, and 120 ms to check *KeyGenAck* and send *KeyGenVerify*. The base station took 90 ms to check *KeyGenStart* and send *KeyGenAck*, and < 1 ms to check *KeyGenVerify*. We note that virtually all of the time required to generate and send *KeyGenStart* is due to two ECC scalar point multiplications (computing the points rP and rG, as described previously). This computation time can be improved by using a more efficient implementation of the basic ECC operations (e.g., [24]). However, in terms of the overall key setup time, ours is the best using this class of mote devices among the studies that we are aware of.

Most importantly, the key exchange protocol only has to be run once to establish the initial symmetric keys, and once each time the keys expire. The Tmote Sky draws a 1.8 mA current when not receiving or transmitting wirelessly [11]. Based on a conservatively low capacity of 500 mAh for a typical AA battery, the mote can run with the processor constantly working for over 11 days on battery power[4]. Consequently, we feel that a periodic investment of 11.4 s for key exchanges will have minimal effect on battery life.

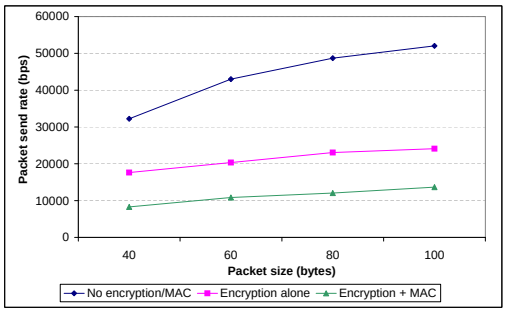

Figure 8. Data throughput as a function of packet size and encryption (the top, middle, and bottom lines correspond to the throughput for no encryption, encryption alone, and encryption with MAC, respectively).

QUERY/DATA PROTECTION MECHANISM

Queries and data are encrypted using the session keys established through the key exchange protocol. To prevent replay or injection of messages, we use both a *session number* (SN) and a *message sequence number* in query and data messages. The session number increases by a random amount whenever the keys are updated. The base station and the mote also increase their message sequence numbers for each new query or data message. Received messages with an unexpected session number or sequence number are discarded. Note that it is difficult for an outside attacker to guess a session number in the first place.

Due to its customizability and ease of implementation, we chose the RC5 block cipher in ciphertext stealing (CTS) mode for symmetric encryption, using the recommended parameters of 64-bit blocks, 128-bit keys, and 12 rounds. For the hash function we chose SHA-1 [25], which allowed us to save code space by using SHA-1 based algorithms for both the message authentication code function (for integrity checking) and key derivation function. We implemented HMAC-SHA-1 [26] and PBKDF1 [27], respectively.

To get an idea of the performance impact of doing symmetric encryption/MAC on every data packet, we also conducted a simple experiment in which mote A sent 2000 packets over the air as rapidly as possible without causing any packet loss at the base station. We varied the packet size from 40 bytes to 100 bytes (near the upper limit of packet size for our platform) and measured the throughput of the system. As illustrated in Fig. 8, using encryption and integrity checking does lower the throughput of the system, from 31-53 kbps to 8-14

[4]We note that the mote's batteries would have to be used to power the attached biometric scanner and medical sensors in addition to the processor. However, the scanner is needed only for initial authentication. Meanwhile, the medical sensors do not have to be continuously working; they might operate in a low-power sleep mode, being activated only once every several seconds to sample the patient's data. We acknowledge, however, that more real-world tests are needed to confirm the feasibility of this setup.

kbps (depending on packet size). However, we feel that this is a reasonable price to pay for the added security. Moreover, the lower send rate is sufficient for a practical deployment. With a packet size of 100 bytes, for example, a send rate of 14 kbps allows a mote to send over 17 packets per second. In a real deployment, the sending rate is likely to be much lower, as energy efficiency concerns would dictate that sensor data be cached on the mote and sent in batches, perhaps at 1 packet per second or less. This would allow a single base station to receive data from 17 patients' motes.

To quantify the ability of a single base station with one attached receiving mote to handle multiple patients, we modified the setup of Fig. 7 to include more than one sending mote, and repeated the previous experiment. For two sending motes, we observed a system bandwidth of about 50, 47, and 28 kbps for the cases of no encryption, encryption alone, and encryption + MAC. The lack of any bandwidth improvement going from 1 to 2 senders in the no encryption case suggests that the serial connection between mote B and the PC is only capable of sustaining about 50 kbps. However, this should be sufficient to support a realistic deployment, as mentioned above. Note that each base station can have multiple attached receiving motes, and multiple base stations can be deployed to support more patients.

RELATED WORK

Several research projects have been developing prototype medical sensor networks, but providing a comprehensive security solution for medical sensor networks remains an open problem. The CodeBlue [4] project at Harvard has proposed a sensor network platform and developed an operational prototype for use in hospitals. The CodeBlue designers acknowledge the need for security in a medical environment, but addressing security requirements is not a main focus of their study. They did develop an ECC implementation on Crossbow motes [28], which is much less efficient than later implementations such as TinyECC.

The I-Living project [29] and PAS [30] propose an architecture to enable assisted living at home for elderly citizens. Although they propose that the medical sensors use IEEE 802.11 or Bluetooth for wireless communication, in contrast to the low-power radios usually found on motes, the aims of the project are very similar. The authors realize the need for privacy when dealing with patient data and propose a symmetric security scheme, in which security information such as keys is stored in USB sticks that are automatically recognized once plugged into a device. However, the scheme seems to require that each device be individually configured.

ALARM-NET [6] shares many similarities with the above work, aiming to develop an architecture for wireless monitoring of residents in assisted-living facilities. ALARM-NET combines wearable medical sensors like those in CodeBlue with static environmental sensors for measuring quantities such as temperature and light. Here, the authors propose that security be provided via the symmetric Advanced Encryption Standard (AES) cipher, but they are not specific about key management.

Much of the aforementioned work has made impressive progress from a hardware perspective, including development of custom medical sensors or integration of sensors with motes and other communication devices. Security has generally been covered separately and not as an integrated component of the architecture. We feel our work is complementary to these efforts, as we have focused on addressing security concerns from an architectural standpoint.

There has been extensive research on wireless sensor security in general. Instead of enumerating all this prior work, we highlight several classes of approaches and explain why they cannot be directly applied here. Efficient public-key based schemes have been proposed for mutual authentication between motes and base stations as well as the establishment of shared keys, e.g. [31, 32]. However, these schemes often require each mote to have its own public-key, which is vulnerable to the physical compromise of motes. Another efficient approach to authentication is to use symmetric cryptography but delay the disclosure of the symmetric keys, e.g. μTESLA [33]. This approach cannot be used in our system, because message authentication has to be instantaneous to ensure timely reaction to emergencies. Thus, we developed our own two-tier authentication scheme to account for the unique characteristics of medical sensor networks.

A large body of work has been done on key pre-distribution using symmetric cryptography (e.g., [34]). These schemes focus mainly on generating shared keys between sensor nodes. Often a pre-configured secret key is used between the base station and each mote, which is not resilient against physical compromise. We chose ECC and asymmetric cryptography as the basis of our key exchange protocol because recent work [28, 21, 12, 24] has shown that performing ECC public-key computations on resource-constrained devices is viable. Several ECC-based key exchange protocols have been proposed, including ECDH [19, 35] and ECMQV [35]. The authenticated versions of ECDH and ECMQV both require a static public key on each mote, so they are not suitable in our system.

CONCLUSION

We have presented the design and implementation of a comprehensive security solution for medical sensor networks. Our initial results on the patient authentication using ECG data are promising, while our implementation on the Tmote Sky demonstrates reasonable computational overhead for our mechanisms. Future plans include improving the neural network performance and optimizing our ECC code, as well as integrating motes with actual medical sensors and fingerprint readers. In addition, we hope to address more security threats such as physical compromise of base stations and denial-of-service attacks. Ultimately, we hope to conduct a study on actual patients to demonstrate the real-world feasibility of our system.

REFERENCES

1. M. Morris, S. S. Intille, and J. S. Beaudin, "Embedded assessment: Overcoming barriers to early detection with pervasive computing," in *Proc. of PERVASIVE 2005*, H. W. Gellersen, R. Want, and A. Schmidt, Eds. Springer-Verlag,

2005, pp. 333–346.

2. S. Stern and D. Tzivoni, “Early detection of silent ischaemic heart disease by 24-hour electrocardiographic monitoring of active subjects,” *British Heart Journal*, vol. 36, pp. 481–486, 1974.
3. R. Fischer, L. Ohno-Machado, D. Curtis, R. Greenes, T. Stair, and J. Guttag, “SMART: Scalable medical alert response technology,” in *Smart Medical Technologies Summit (SMT)*, 2004.
4. V. Shnayder, B.-R. Chen, K. Lorincz, T. R. F. Fulford-Jones, and M. Welsh, “Sensor networks for medical care,” Harvard University, Tech. Rep. TR-08-05, Apr. 2005.
5. C. Park, P. H. Chou, Y. Bai, R. Matthews, and A. Hibbs, “An Ultra-Wearable, Wireless, Low Power ECG Monitoring System,” *Proceedings of IEEE BioCAS*, Nov. 2006.
6. A. Wood, G. Virone, T. Doan, Q. Cao, L. Selavo, Y. Wu, L. Fang, Z. He, S. Lin, and J. Stankovic, “ALARM-NET: Wireless Sensor Networks for Assisted-Living and Health Monitoring,” University of Virginia, Tech. Rep. CS-2006-01, 2006.
7. T. Gao, C. Pesto, L. Selavo, Y. Chen, J. Ko, J. Lim, A. Terzis, A. Watt, J. Jeng, B.-R. Chen, K. Lorincz, and M. Welsh, “Wireless medical sensor networks in emergency response: Implementation and pilot results,” in *Proc. 2008 IEEE Int. Conf. Technologies for Homeland Security*, Waltham, MA, 2008.
8. Crossbow Technology, “MPR/MIB mote hardware users manual,” Jan. 2006, `http://www.xbow.com/Support/manuals.htm`.
9. Office for Civil Rights, United State Department of Health and Human Services, “Medical Privacy - National Standards to Protect the Privacy of Personal Health Information,” `http://hhs.gov/ocr/hipaa/finalreg.html`.
10. K. K. Venkatasubramanian and S. K. S. Gupta, “Security solutions for pervasive healthcare,” in *Security in Distributed, Grid, Mobile, and Pervasive Computing*, Y. Xiao, Ed., 2007.
11. Moteiv Corporation, “Tmote Sky,” 2007, `http://www.moteiv.com/products/tmotesky.php`.
12. N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, “Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs,” in *Workshop on Cryptographic Hardware and Embedded Systems*, Aug. 2004.
13. D. Chaum and E. van Heijst, “Group Signatures,” in *Advances in Cryptology - Eurocrypt ’91*, 1991, pp. 257–265.
14. “Fujitsu MBF200 Solid State Fingerprint Sensor,” `http://www.fujitsu.com/emea/services/microelectronics/sensors/`.
15. ODI Security, “Embedded Fingerprint Matching Module Utilizing Fujitsu Array Sensor,” `http://www.odisecurity.com/`.
16. A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, “PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals,” *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000 (June 13).
17. N. Koblitz, “Elliptic curve cryptosystems,” *Mathematics of Computation*, vol. 48, pp. 203–209, 1987.
18. V. Miller, “Use of elliptic curves in cryptography,” in *CRYPTO 85*, 1985.
19. Certicom Research, “Standards for Efficient Cryptography (SEC) 1: Elliptic Curve Cryptography,” Sept. 2000.
20. “TinyOS Website,” `http://www.tinyos.net/`.
21. A. Liu, P. Kampanakis, and P. Ning, “TinyECC: Elliptic Curve Cryptography for Sensor Networks,” `http://discovery.csc.ncsu.edu/software/TinyECC/`.
22. S. C. Shantz, “From Euclid’s GCD to Montgomery Multiplication to the Great Divide,” Sun Microsystems, Tech. Rep. TR-2001-95, June 2001.
23. Certicom Research, “Standards for Efficient Cryptography (SEC) 2: Recommended Elliptic Curve Domain Parameters,” Sept. 2000.
24. H. Wang, B. Sheng, C. C. Tan, and Q. Li, “WM-ECC: an Elliptic Curve Cryptography Suite on Sensor Motes,” Dept. of Computer Science, College of William and Mary, Tech. Rep. WM-CS-2007-11, 2007.
25. D. Eastlake and P. Jones, “US Secure Hash Algorithm 1,” Sept. 2001, RFC 3174, `http://www.ietf.org/rfc/rfc3174.txt`.
26. H. Krawczyk, M. Bellare, and R. Canetti, “HMAC: Keyed-Hashing for Message Authentication,” Feb. 1997, RFC 2104, `http://www.ietf.org/rfc/rfc2104.txt`.
27. B. Kaliski, “PKCS #5: Password-Based Cryptography Specification,” Sept. 2000, RFC 2898, `http://www.ietf.org/rfc/rfc2898.txt`.
28. D. J. Malan, M. Welsh, and M. D. Smith, “A Public-Key Infrastructure for TinyOS Based on Elliptic Curve Cryptography,” in *Proceedings of the IEEE International Conference on Sensor and Ad Hoc Communications and Networks*, Oct. 2004.
29. Q. Wang, W. Shin, X. Liu, Z. Zeng, C. Oh, B. Al-Shebli, M. Caccamo, C. Gunter, E. Gunter, J. Hou, K. Karahalios, and L. Sha, “I-Living: An open system architecture for assisted living,” in *Proceedings of the IEEE SMC*, 2006.
30. J. C. Hou, et al., “PAS: A wireless-enabled, sensor-integrated personal assistance system for independent and assisted living,” *Proc. of Joint Workshop on High Confidence Medical Devices, Software, and Systems (HCMDSS) and Medical Device Plug-and-Play (MD PnP) Interoperability (HCMDSS/MD PnP’07)*, June 2007.
31. M. Aydos, B. Sunar, and C. K. Koc, “An elliptic curve cryptography based authentication and key agreement protocol for wireless communication,” in *Proceedings of the 2nd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 1998.
32. Q. Zhang, J. Cukier, H. Kobayashi, B. Liu, and J. Zhang, “Fast authenticated key eastblishment protocols for self-organizing sensor networks,” in *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications*, 2003, pp. 141–150.
33. A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, “SPINS: Security protocols for sensor networks,” in *Proceedings of the ACM MOBICOM*, 2001.
34. S. Zhu, S. Setia, and S. Jajodia, “LEAP: Efficient security mechanisms for large-scale distributed sensor networks,” in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, 2003.
35. NIST, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, Special Publication 800-56A,” 2007, `http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf`.