# Addressing Security in Medical Sensor Networks*

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

## ABSTRACT

We identify the security challenges facing a sensor network for wireless health monitoring, and propose an architecture called "SNAP" (Sensor Network for Assessment of Patients) to address these challenges. SNAP protects the privacy, authenticity, and integrity of medical data, with low-cost energy-efficient mechanisms. We have incorporated the following mechanisms in SNAP: (1) an ECC-based secure key exchange protocol to set up shared keys between sensor nodes and base stations; (2) symmetric encryption and decryption for protecting data confidentiality and integrity; (3) a two-tier authentication scheme for verifying data source. We have developed a prototype on the Tmote Sky platform for evaluating the proposed architecture and mechanisms.

## Categories and Subject Descriptors

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

## General Terms

Security, Human Factors

## Keywords

medical sensor networks, elliptic curve cryptography, Tmote Sky

## 1. INTRODUCTION

In our aging society, an increasing number of people have chronic medical conditions such as diabetes and heart disease. If these people's health conditions could be monitored continuously and remotely, medical professionals could react to life-threatening situations such as heart attacks much more quickly. Moreover, since each patient's data is collected over a long period of time, physicians could provide more accurate diagnoses and better treatment. Current monitoring solutions, however, are both cumbersome and costly. Typically, a patient is attached to a number of medical sensors that convey information on his or her vital signs to a bedside monitoring device. However, because these connections are wired, such a setup severely limits the mobility of the patient, making it unsuitable for long-term continuous health monitoring.

One potential solution is to attach small, lightweight wireless medical sensors to patients. Several research groups (e.g. [8, 20, 17, 22]) have recently integrated medical sensors with wireless motes for health monitoring, such as the Harvard wireless pulse oximeter [20]. These medical sensors wirelessly transmit data to physicians and nurses. This frees the patient from the confinement of traditional wired sensors, allowing him/her to move about at leisure and increasing comfort. Such medical sensor networks can be deployed in *hospitals, long-term care facilities, and homes.*

Our work addresses the security challenges in medical sensor networks. We believe that security must be designed into the architecture from day one rather than being added 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.

Existing sensor network research has mainly focused on monitoring the *physical* environment. However, a medical sensor network monitors *humans.* 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, leading to these security challenges:

- How to ensure the privacy and integrity of the medical data, given that the wireless channel is easily subject to many forms of attacks?
- How to ensure that only authorized people can access the data? The solution should scale to a large number of users (medical professionals and patients) and accommodate changes in the users.
- How to prevent someone from using captured sensors to recover sensitive medical information or inject false information? Attackers have relatively easy physical access to these sensors since they are located on patients. Moreover, the sensors may simply get lost while the patients are moving around.

*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.


*HealthNet'07*, June 11, 2007, San Juan, Puerto Rico, USA.

The above scenarios may seem somewhat paranoid, but we have to foresee the possible attack scenarios, as legal and ethical considerations mandate that medical data be kept private. Users will not deploy this system if they are not convinced that their data will be kept confidential, regardless of how good the system's performance is.

What makes securing sensor networks more difficult than other types of networks is that wireless sensor nodes usually have limited resources, while conventional security mechanisms incur high costs in terms of CPU, memory, bandwidth, and energy consumption. For example, the RSA public-key scheme requires storing keys longer than 1 KB (for reasonable security) in memory, while the popular Crossbow MICA motes [6] have a mere 4 KB of RAM. Sensor nodes also have much more limited processing power than PCs or PDAs; MICAs, for example, use an 8-bit, 8 MHz processor, and the comparable Moteiv Tmote Sky platform [16] employs a 16-bit, 8 MHz processor. The recent Intel iMote platform has higher processor speed and more RAM, but it consumes much more energy than the previously mentioned motes.

The contribution of our work is two-fold. First, we propose an architecture called "SNAP" (Sensor Network for Assessment of Patients) and its associated mechanisms for securing medical sensor networks. More specifically, SNAP protects the privacy, authenticity, and integrity of medical data, with low-cost energy-efficient mechanisms. We have incorporated the following mechanisms in SNAP: (1) an ECC-based secure key exchange protocol to set up shared keys between sensor nodes and base stations; (2) symmetric encryption and decryption for protecting data confidentiality and integrity; (3) a two-tier data source authentication scheme based on patient biometric and medical data. Second, we have developed a prototype on the Tmote Sky platform for evaluating the security, cost, and performance of the proposed architecture and mechanisms.

In the remainder of this paper, we first discuss the security threats and challenges in §2. In §3 and §4, we present our SNAP architecture and its associated mechanisms. We briefly describe our current implementation and present preliminary results in §5. We present related work in §6 and conclude our paper in §7.

## 2. SECURITY THREATS/CHALLENGES

In this section, we discuss the potential security threats to medical sensor networks and then point out what makes addressing these threats challenging. We assume that the sensor network contains some sensor nodes that collect data from patients, one or more base stations to receive the data from the sensor nodes, and some relay nodes that deliver the data from the sensor nodes to the base stations.

We classify potential security threats into two categories: outsider attacks and insider attacks. *Outsider attacks* are perpetrated by attackers who do not have control of a valid sensor node, base station, or any other nodes in the network. These attacks include, but are not limited to, the following: (1) eavesdropping on data; (2) spoofing of a base station to receive sensor data; (3) replay of previous queries to obtain sensor data; (4) modification or injection of data without the knowledge of the source or destination; (5) spoofing of a sensor to report forged data; and (6) replay of previous data. Among these attacks, (1)-(3) compromise the privacy of patient data, while (4)-(6) compromise the authenticity and integrity of patient data.

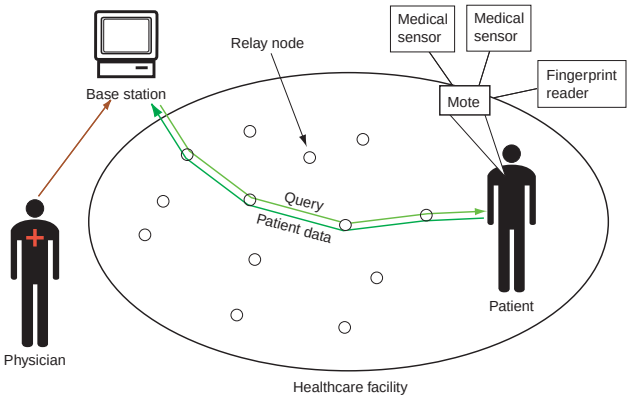


**Figure 1: SNAP Architecture.**

*Insider attacks* are launched by attackers who have control of some nodes in the network. If the attacker controls a sensor node, s/he can easily forge data that appears to be from a legitimate patient. The attacker can also obtain any existing cached data in the node. If the attacker controls a base station, s/he can access the private data from the sensors. Insider attacks are much more difficult to handle, since other nodes cannot distinguish the attacker from a legitimate node by the use of shared secrets (by compromising a node, the attacker has access to these secrets).

Similar threats exist in traditional ad hoc wireless networks and other types of sensor networks. However, addressing these threats in a medical sensor network poses several unique challenges. First, there is a stringent expectation for the system to ensure the privacy of medical data. Second, sensor nodes have much lower processor speed, memory, link bandwidth, and energy supply than mobile PCs or PDAs, so the security mechanisms must be resource-efficient. Third, due to their small size, the individual sensors can be easily stolen or simply lost. Patient mobility also makes it more likely for patients to lose sensors (most other sensor networks have stationary sensors). Furthermore, attackers can easily find their targets in local hospitals or healthcare facilities, as opposed to remote locations like forests or battlefields. Hence, physical compromise of sensor nodes is more likely in medical sensor networks than in other types of sensor networks. Fourth, there may be many users (physicians, nurses, patients) who are authorized to access the data, so a scalable solution to authenticate the users must be provided to ensure the privacy of the data.

## 3. SNAP ARCHITECTURE

In this section, we first describe the overall SNAP architecture. We then discuss our architectural choices and their effect on security.

### 3.1 Architecture Components

In SNAP (Figure 1), each *patient* has one wireless mote attached to his/her body. The mote is connected to several *medical sensors*, which take samples of the patient's health data when they are activated. The main tasks of the mote are the following: (a) authenticate the patient with the *base station* using our two-tier authentication system (§4.1); (b) establish a symmetric key with the base station using our ECC-based secure key exchange protocol (§4.2);

and (c) communicate with the base station to receive queries and send sensor data, both of which are encrypted on an end-to-end basis using a symmetric key.

A *medical professional* issues queries for a patient's data through one or a few *base stations*. As the base stations must communicate with the motes, they must be located in the same physical area as the patients. However, the base stations may be accessed either directly or remotely (e.g., from a physician's home) for ease of patient monitoring. 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 sensor is deactivated by the medical professional or the patient).

Wireless *relay nodes* throughout the healthcare facility forward the queries and patient data between the base stations and the motes. In an indoor environment, the relay nodes could have a continuous power supply, so they can be powerful motes such as the Intel iMote.

Note that the labels of Figure 1 (*patient*, *healthcare facility*, etc.) are only for illustrative purposes. The same architecture could be applied, for example, to monitor sickly people in their homes.

## 3.2 Architectural Choices for Enhanced Security

We have made several explicit architectural choices to enhance security. These choices differentiate SNAP from previously proposed architectures such as the CodeBlue architecture described in [20].

First, *the motes communicate only with the base stations, not individual users' computers*. This is because in a large healthcare facility, it would be difficult for the motes to authenticate hundreds of individual users (physicians, nurses, staff, etc.). Each mote would have to maintain complete access control information about users' privileges and their identity information (e.g., public keys or passwords). The motes simply do not have the memory resources to maintain all this information. Moreover, it would be infeasible to update the access control information in every mote whenever a change in user privileges is required. Therefore, we decide to authenticate the users at the base stations and have the base stations issue the queries to the motes on behalf of the users. Motes are configured with the base stations' addresses and send their data only towards the base stations.

Second, *in order to handle spoofing of base stations, the motes are also configured with the base stations' public keys so that they can establish symmetric keys only with valid base stations*. If there are many base stations in the facility, we can use the *group public-key scheme* proposed in [4]. This scheme allows each base station to have its own private/public key, while allowing the motes to use only the group's public key to authenticate the base stations.

Third, *we do not assume that each mote has its own public/private key pair*. The main reason behind this choice is that individual motes are relatively easy to physically compromise. When a mote possesses a permanent private key and is captured by an attacker, the attacker may be able to derive the symmetric key as well as decrypt the data previously sent by the mote (assuming that the attacker has been eavesdropping on the communication between the mote and the base station).

Fourth, *in order to handle forged data, a base station will not accept data from a mote unless the mote is attached to a patient registered in the system*. Consequently, the base station needs some way of verifying the identity of each patient. We propose a two-tier authentication system based on patient data, discussed in §4.1. As part of this authentication system, we assume that each mote is connected to a small fingerprint scanner or a comparable biometric identication device.

# 4. SNAP SECURITY MECHANISMS

In this section, we describe three security mechanisms in SNAP, with an emphasis on how they protect the communication between motes and base stations against various attacks. Note that we try to minimize the number of computationally intensive operations on the motes.

## 4.1 Two-Tier Data Source Authentication Scheme

An attacker can inject forged 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. At the first tier, each mote is integrated with a small biometric scanner such as a fingerprint reader (similar to [1]) or a finger vein reader, allowing the sensor to capture a unique signature for the patient. This 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, and also that the space of possible signatures is sufficiently large that a brute-force attack is infeasible.

The first tier alone is insufficient to guard against forged data, because an attacker could compromise an authenticated mote and proceed to successfully send forged data to the base station. One approach of preventing this would be to force periodic re-authentication of the mote using the biometric scanner; however, this requires extra bandwidth overhead and is inconvenient for the patient. Instead, we propose implementing a second-tier authentication system that tries to assert the identity of a patient based solely on the sensor data being collected from that patient. Data from the sensor is continually passed to a filter, which determines whether the data "makes sense" for that patient. Such an approach requires statistical or machine learning techniques to recognize medical data as being from a particular patient. For example, electrocardiogram (ECG) signals have been successfully used in [12] and [11] to identify patients.

To use the second-tier authentication system, 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. Note that both the initial learning and the detection mechanism is run on the base station, in order to minimize the computation on the mote.

## 4.2 ECC-based Key Exchange Protocol

As mentioned earlier, efficient operation is important for resource-constrained motes. For efficiency, we encrypt queries and patient data using pairwise *symmetric keys* shared between the base station and motes. However, symmetric keys are difficult to manage and update manually. Hence, we propose a key exchange protocol that each mote uses to securely derive a randomly generated symmetric key with the base station at the beginning of their communication. We use *elliptic curve cryptography* (ECC) [2] 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).

### 4.2.1 Elliptic Curves

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; the elements of this field are the integers in $[0, p-1]$.

### 4.2.2 ECDLP

ECC's security is due to the computational difficulty of the *elliptic curve discrete logarithm problem* (ECDLP). Given two points $G$ and $Q = nG$ on an elliptic curve over a finite field, it is hard to determine $n$. At the same time, $Q$ is fairly easy to determine, given $n$ and $G$. To use ECC, two nodes $A$ and $B$ need to 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 $n$ as its private key and derive its public key $P$ by using $P = nG$ (this is called scalar point multiplication). If $n$ is large, it is hard for an attacker to derive the private key $n$ from the public key $P$ even if the attacker knows $G$, due to the difficulty of ECDLP.

### 4.2.3 ECIES

To protect the messages from node $B$ to node $A$ against eavesdropping and modification, the messages can be encrypted using the *Elliptic Curve Integrated Encryption Scheme* (ECIES) [2] with $A$'s public key $P$. More specifically, 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: an encryption key $K_s^{'}$ and a MAC (message authentication code) key $K_{mac}^{'}$ from $S$. These two keys are used to encrypt the messages from $B$ to $A$. 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 $n$ and $R$: $S = nR = n(rG) = r(nG) = rP$. Node $A$ then uses the same KDF to derive $K_s^{'}$ and $K_{mac}^{'}$ from $S$ and decrypts the message. Again, due to the difficulty of the ECDLP, it is infeasible for an eavesdropper to derive $r$ (and hence $S$) even with knowledge of $R$ and $G$, provided $r$ is sufficiently large.

### 4.2.4 Protocol

In SNAP, each base station has a private key $n_B$ and a public key $P_B = n_B G$, where $G$ is a chosen base point on the elliptic curve. The public key $P_B$ is pre-configured in the motes. Our key exchange protocol, which involves three messages $KeyGenStart$, $KeyGenAck$, and $KeyGenVerify$, is described below (Figure 2). Note that we do **not** use

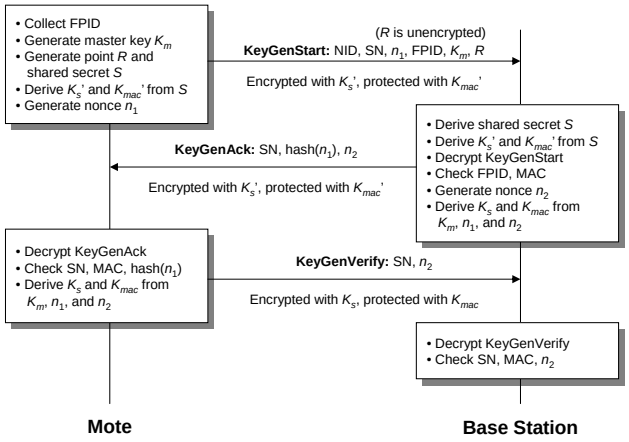


**Figure 2: ECC-based Key Setup Protocol.**

*elliptic curve Diffie-Hellman* (ECDH) key exchange, because it assumes that both parties have public/private key pairs.

When a wireless mote is attached to a patient, the patient uses the fingerprint scanner on the mote to activate the key exchange protocol. 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 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 $n_1$.

The base station decrypts the message 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. These keys can be updated by changing $n_1$ and $n_2$. 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 using the ECIES procedure. The mote decrypts the message 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 integrity-protected with $K_{mac}$ and encrypted with $K_s$. The base station decrypts the message 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.

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. The update messages contain the existing session number $SN$, the new session number $SN'$, and the new nonce values,

encrypted using the existing session keys. The value $SN$ protects against replay attacks, while encryption prevents outside attackers from forging meaningful messages.

#### 4.2.5 Attacks and Countermeasures

To prevent replay attacks, each mote stores its current $SN$ in non-volatile memory and increases $SN$ by a random amount whenever the key exchange protocol or the key update procedure is run. Therefore, an attacker cannot inject previously recorded *KeyGenStart*, *KeyGenAck*, and $KeyGenVerify$ messages in a later session. Moreover, the $FPID$ helps identify forged *KeyGenStart* messages, even if an outside attacker is able to guess a reasonable $SN$. Furthermore, all three messages are encrypted, making it difficult for an outside attacker to obtain $FPID$, $K_m$, $SN$, $n_1$, and $n_2$. Finally, we include two additional checks to further thwart forged messages. The *KeyGenAck* message contains a hash of $n_1$, so an outside attacker must guess this hash correctly. The $KeyGenVerify$ message is encrypted and integrity-protected with the session keys and contains the value $n_2$, so it is impossible for an outside attacker to forge this message unless s/he guesses all these values correctly.

We use the following measures to limit what an inside attacker can do with a compromised mote. First, the $FPID$ is erased from the mote's memory once the *KeyGenAck* message is received, so it is difficult for an attacker to obtain the previous patient's fingerprint information. Second, $n_1$ and $n_2$ are erased as soon as the session keys are derived from them. Third, once new session keys are derived, the previous session's keys are erased from memory. Fourth, if the base station and the mote have not communicated for a long period of time, the master key $K_m$ and its derived keys expire and are removed from memory. These measures limit how much private data the attacker can recover using pre-captured data for that mote.

### 4.3 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 $SN$ and a *message sequence number* in query and data messages. The base station and the mote increase their sequence numbers for each new query or data message; received messages with an unexpected $SN$ or sequence number are discarded. Note that it is difficult for an outside attacker to guess $SN$ in the first place. For an inside attacker with a compromised mote, s/he may be detected by the base station, which monitors the patient data to detect forgeries (see §4.1).

## 5. IMPLEMENTATION AND EVALUATION

We have implemented the key exchange protocol and symmetric data encryption mechanism in TinyOS 2 on the Moteiv Tmote Sky platform, which features a 16-bit, 8 MHz Texas Instruments MSP430 processor with 48 KB of program ROM and 10 KB of RAM.

Our ECC implementation is partly based on NCSU's TinyECC code for the Crossbow MICA [15]. 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 [19]. We then implemented ECIES for secure key exchange, using the elliptic curve secp160r1 defined over a 160-bit prime field as recommended by [3]. We use 160-bit private keys, 320-bit public keys, and a 160-bit random number in ECIES.

We chose the RC5 block cipher for symmetric encryption, using the recommended parameters of 64-bit blocks, 128-bit keys, and 12 rounds. For the hash function we use SHA-1, based on a reference implementation in RFC 3174 [7]. To save code space, we use SHA-1 based algorithms for both the message authentication code function (for integrity checking) and key derivation function; we implemented HMAC-SHA-1 [14] and PBKDF1 [13], respectively.

For the key exchange protocol, we use a 32-bit session number ($SN$), 32-bit patient ID ($FPID$), 128-bit master key ($K_m$), 128-bit transient and session keys ($K_s$, $K_s^{'}$, $K_{mac}$, $K_{mac}^{'}$), and 64-bit nonces ($n_1$, $n_2$). The resulting *KeyGenStart*, *KeyGenAck*, and $KeyGenVerify$ messages are 54, 52, and 32 bytes long.

Our entire code on the Tmote uses 24.1 KB of ROM and 2.75 KB of RAM, which leaves ample room for other applications. The ECC code is quite efficient – it carries out a 160-bit scalar point multiplication (the most computationally intensive operation in ECC), in 5.3 s. We timed the protocol running on two motes, one serving as a sender and one serving as a "base station." The sending mote took 11.4 s to generate and send *KeyGenStart*, and 120 ms to check *KeyGenAck* and send *KeyGenVerify*. The base station took 6.86 s to check *KeyGenStart* and send *KeyGenAck*, and 32.8 ms to check *KeyGenVerify*. The choice of using a mote for a base station was done mainly for ease of development, so that the cryptographic operations written for the mote could be reused. However, we note that in a real deployment, the base station's computations would be performed on a PC-class device, resulting in much faster numbers. A mote would be needed only to serve as the interface between the sensor network and the base station.

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 one mote sent 1000 packets as rapidly as possible to a receiving mote, which forwarded the packets to a PC via a serial interface (emulated in USB). We varied the packet size from 40 bytes to 100 bytes in different experiments. We then measured the raw sending rate and packet loss as seen from the PC. Due to space constraints, the figures are not shown here. We observed that using encryption does lower the sending rate of the system, from 4-7 KB/s to 1-2 KB/s (depending on packet size). However, we also observed significant packet loss without encryption, on the order of 20-40%, while the experiments with encryption did not experience any losses. Although we are still investigating the causes, one possibility is that the serial connection between the receiving mote and the PC could not handle the faster send rate without encryption. In summary, the experiment suggests that our system is capable of handling sampling rates below 20 Hz with a data packet size of 100 bytes. In a real deployment, the sampling rate is likely to be much lower than 20 Hz. Nevertheless, we believe that we can further improve the efficiency of the symmetric encryption implementation.

## 6. RELATED WORK

Early work on sensor network security focused on symmetric cryptography (e.g., [18]) for protecting sensitive data. However, symmetric schemes themselves do not provide a

mechanism for key distribution and management; this issue is addressed by public-key cryptography. Recent research (e.g., [15, 9, 10]) has shown that performing public-key computations on resource-constrained devices is viable.

Despite this progress, providing a comprehensive security solution for medical sensor networks remains an open problem. The designers of CodeBlue [20] acknowledge the need for security in a medical environment, but their work does not focus on addressing security requirements. The MUSIC project [5] proposes a three-tier architecture for patient health monitoring, but it focuses largely on optimizing network lifetime by controlling the frequency at which data is collected and transmitted. I-Living [21] proposes an architecture to enable assisted living at home for elderly citizens. They realize the need for security when dealing with patient data and propose a symmetric security scheme, in which security context information such as keys and certificates are stored in USB sticks that are automatically recognized and read once plugged into a device. However, the scheme seems to require that each device be individually configured. In ALARM-NET [22], the authors propose that security be provided via the symmetric Advanced Encryption Standard (AES) cipher; however, they are not specific about how key management is to be accomplished. Our work is complementary to the above efforts as our ECC-based key exchange protocol addresses the need for key management and our two-tier data authentication scheme makes it difficult for attackers to inject forged data.

# 7. CONCLUSION

We have presented the SNAP architecture for medical sensor networks and its associated security mechanisms. We are optimizing our current implementation and adding an ECG-based mechanism to authenticate patients. We will perform more extensive evaluation of our implementation and release our code in the near future. In addition, we will develop more realistic base station software, in which a mote serves only as the interface between the sensor network and a PC responsible for cryptographic operations/storage of received data. Once sufficient security measures are in place, we will address routing, mobility, and congestion issues. We also plan to develop a hardware prototype using actual medical sensors and perform preliminary field trials in local hospitals.

# 8. REFERENCES


[1] Fujitsu MBF200 Solid State Fingerprint Sensor. http://www.fujitsu.com/emea/services/microelectronics/sensors/.

[2] Certicom Research. Standards for Efficient Cryptography (SEC) 1: Elliptic Curve Cryptography. Sept. 2000.

[3] Certicom Research. Standards for Efficient Cryptography (SEC) 2: Recommended Elliptic Curve Domain Parameters. Sept. 2000.

[4] D. Chaum and E. van Heijst. Group Signatures. In *Advances in Cryptology - Eurocrypt '91*, pages 257–265, 1991.

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

[6] Crossbow Technology. MPR/MIB mote hardware users manual. http://www.xbow.com/Support/manuals.htm, Jan. 2006.

[7] D. Eastlake and P. Jones. US Secure Hash Algorithm 1. RFC 3174, http://www.ietf.org/rfc/rfc3174.txt, Sept. 2001.

[8] 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.

[9] 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.

[10] H. Wang and B. Sheng and Q. Li. TelosB implementation of elliptic curve cryptography over primary field. Technical Report WM-CS-2005-12, Dept. of Computer Science, College of William and Mary, Oct. 2005.

[11] H. G. Hosseini, D. Luo, and K. J. Reynolds. The Comparison of Different Feed Forward Neural Network Architectures for ECG Signal Diagnosis. *Medical Engineering and Physics*, 28:372–378, 2006.

[12] S. A. Israel, J. M. Irvine, A. Cheng, M. D. Wiederhold, and B. K. Wiederhold. ECG to identify individuals. *Pattern Recognition*, 38:133–142, 2005.

[13] B. Kaliski. PKCS #5: Password-Based Cryptography Specification. RFC 2898, http://www.ietf.org/rfc/rfc2898.txt, Sept. 2000.

[14] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, http://www.ietf.org/rfc/rfc2104.txt, Feb. 1997.

[15] A. Liu, P. Kampanakis, and P. Ning. TinyECC: Elliptic Curve Cryptography for Sensor Networks. http://discovery.csc.ncsu.edu/software/TinyECC/.

[16] Moteiv Corporation. Tmote Sky. http://www.moteiv.com/products/tmotesky.php, 2007.

[17] 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.

[18] 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.

[19] S. C. Shantz. From Euclid's GCD to Montgomery Multiplication to the Great Divide. Technical Report TR-2001-95, Sun Microsystems, June 2001.

[20] V. Shnayder, B.-R. Chen, K. Lorincz, T. R. F. Fulford-Jones, and M. Welsh. Sensor networks for medical care. Technical Report TR-08-05, Harvard University, Apr. 2005.

[21] 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.

[22] 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. Technical Report CS-2006-01, University of Virginia, 2006.