A Secure Protocol for Ubiquitous Sensing and Monitoring of Patient Conditions

Eric Sabbah
City University of New York, The College of Technology
and
Kyoung-Don Kang
State University of New York at Binghamton

Wireless sensor devices are envisioned to collect long term medical data that aid medical professionals in the care and treatment of the elderly and other patients with chronic diseases. Using wearable body sensor networks (BSNs), patients may live more mobile and independent lives than they would in a hospital, nursing home, or rehabilitation center. As a result, the quality of life can be improved and the cost for medical care can be reduced. Although security issues in BSNs have been studied previously, most existing work focuses on monitoring patients in a fixed location such as a hospital and nursing home, relying on centralized approaches that often have poor scalability. In this paper, we develop a new protocol to support secure and scalable anytime/anywhere monitoring of patients to give them full mobility, support the security of sensitive medical data, and preserve the location privacy of patients, while sharing sensor data collection and retrieval workloads among distributed storage nodes.

Keywords: WSN, BSN, Privacy, Scalability, IBE, Authentication, Authorization, Ubiquitous Computing

1. INTRODUCTION

Advances in the development of wireless technology, physiological sensors, and low power computing have led to the availability of modern wireless sensor networks (WSNs)– see Table I. These systems allow for rapid data collection in situations where human observation would be undesirable, inefficient, expensive, or dangerous. They are invaluable to many applications, such as environmental observation, assisted living, disaster recovery, and battlefield monitoring [Sabbah et al. 2008].

In this paper, we focus on secure sensing and monitoring of medical conditions of patients with chronic diseases. The direct health care costs of chronic conditions account for approximately 75% of U.S. health care expenditures according to the Center for Disease Control and Prevention [cdc-overview 2010]. Further, 133 million Americans−nearly one out of two adults−suffer at least one chronic disease. The increasing number of chronic conditions often result in decreased functionality, decreased physical wellbeing [Patrick et al. 2000; Stewart et al. 1989], increased mortality, decreased quality of life, and increased use of health services [Gijsen et al. 2001].

As an effort to improve the quality of a patient's life and reduce the cost for chronic disease care, we develop a new protocol to support anytime/anywhere collection, storage, and retrieval of sensor data in a secure, scalable manner. We assume that medical sensor data can be collected by using wireless body sensor networks (BSNs) – see Table I – that are available, for example, in wearable form. A BSN comprised of wireless sensor devices can monitor health conditions on a continuous, efficient, and long term basis, while allowing the patients to continue to maintain the dignity of living an independent, ambulatory life in their own home [Yang 2006].[1]

Once sensor data are collected by the wearable nodes, an efficient mechanism is required to deliver them to appropriate readers, such as doctors, nurses, and others involved in the health care industry.

[1]Developing a BSN is not the focus of this paper. Assuming the availability of BSNs, we aim to develop a scalable security protocol in this paper.

Authors' addresses: Eric Sabbah, CUNY, The College of Technology, 300 Jay Street, Office Namm 912, Brooklyn, NY 11201
Kyoung-Don Kang, Binghamton University, State University of New York, PO Box 6000, Binghamton, NY 13902-6000

Ideally, information needs to be accessed anywhere, anytime, and in any quantity−subject to the storage capacity and reader authorization. However, relatively little work has been done to securely store and retrieve sensitive sensor data, e.g., the electrocardiogram (EKG)– see Table I – data of a patient with a heart disease, in a distributed environment such as the Internet. Instead, most existing work on wireless medical sensing, such as [Malan et al. 2004; Lo and Yang 2005; Ganti et al. 2006; Zhong et al. 2006; Patel et al. 2008; Gao et al. 2007; Malasri and Wang 2007; Mont et al. 2003; Tan et al. 2008], has focused on monitoring patients in a hospital, nursing home, or other fixed/concentrated location. In these approaches, data are stored at the BSN nodes themselves or at a centralized storage site. Unfortunately, these relatively centralized approaches confine patients to hospitals or other medical institutions. Also, they may suffer the lack of scalability. Despite the importance, relatively little work has been done to support scalable as well as secure ubiquitous sensing and monitoring of patients.

In this paper, we aim to improve the flexibility and scalability of secure patient monitoring. A summary of our key contributions follows:

— We design a new protocol to support secure anytime/anywhere sensing and monitoring of patients. Neither patients nor medical professionals are confined to a hospital or any other medical institution. As a result, the flexibility of patient monitoring is enhanced significantly. Also, the quality of life could be improved.

— To develop a scalable security protocol, we apply efficient cryptographic techniques such as the identity based encryption (IBE) algorithm [Shamir 1985] and Elliptic Curve Cryptography (ECC) [Hankerson et al. 2004; Menezes et al. 1996] – see Table I. They substantially reduce the complexity of cryptographic key management and overhead of public key systems.

— By employing distributed storage nodes, our protocol collects, stores, and retrieves sensor data in a scalable and secure manner compared to centralized approaches. Notably, one cannot simply add more storage nodes to a centralized storage system to support secure anytime/anywhere sensing. In fact, the naive approach may introduce a lot of security and privacy issues. To avoid these problems, we consider the security and privacy requirements for ubiquitous sensing and monitoring of patients throughout the protocol design process, while addressing the aforementioned scalability and flexibility limitations of most existing approaches.

— The privacy of sensitive medical data are supported by storing data as encrypted, while keeping the private keys needed to decrypt the data separately from the data storage nodes. Even if an adversary compromises a data storage node, he cannot decrypt the data. Moreover, due to the scalable as well as secure design, a system administrator can add more storage nodes, for example, to reduce the medical data service delay, if the demand for service increases without introducing security problems.

— By applying efficient one-way hashing techniques, we provide load balancing among the storage nodes to improve the scalability. At the same time, we support the location privacy of patients by making it difficult for an adversary to associate the location of the storage node selected to save a patient's medical data during a certain period of time with the actual location of the patient.

— Moreover, we have actually implemented our protocol and evaluated its overhead and performance. The resource requirements of our protocols fall well within the finite resource limits provided by handheld devices used to collect and access medical sensor data. Also, our protocol generally scalable, while supporting the desired security and privacy features described before.

The rest of the paper is organized as follows. Section 2 discusses related work. A detailed description of our protocol is given in Section 3. In Section 4, we discuss implementation details and evaluate the performance. Finally, Section 5 concludes the paper and outlines future work.

2. RELATED WORK

WSN security issues have been studied thoroughly in the literature including [Perrig et al. 2001; Karlof et al. 2004; Eschenauer and Gligor 2002; Chan et al. 2003; Du et al. 2005; Čapkun et al. 2003; Liu and Ning 2008; 2003; Malan et al. 2004; Sabbah et al. 2008]. However, most of existing approaches to WSN security mainly deal with security *within a WSN only*. Also, most of them rely on relatively weak security

Acronym	Meaning
BSN	body sensor network
EKG	electrocardiogram
WSN	wireless sensor network
AP	access point
CA	certificate authority
DS	data store
IBE	identity-based encryption
ECC	elliptic curve cryptography
PDA	personal digital assistant
ID	data identifier used by IBE
$H_{storage}$	hash function used to select DS
H_{IBE}	hash function used to derive keys in IBE
MAC	message authentication code
TTP	trusted third party
RSA	Rivest, Shamir and Adleman's public key scheme
SSL	secure sockets layer
TLS	transport layer security

Table I. Table of acronyms Used in this paper.

mechanisms due to the resource constraints in WSNs. Therefore, they are not directly applicable to the problem of supporting secure, scalable sensing and monitoring addressed in this paper.

For secure sensor data collection and retrieval, we apply the concept of identity base encryption (IBE) that was first proposed by [Shamir 1985] and substantially extended by [Boneh and Franklin 2001] and [Cocks 2001]. IBE refers to a category of cryptographic systems where certificate set-up and public key infrastructure deployment is significantly simplified without compromising security. It allows for great flexibility, because a user can derive a public key from a string whose format is known to communication parties in advance. These strings generally include meta-data about the information that is being encrypted/decrypted. For example, a user, John, may send an encrypted email to Mary using a string: mary@hotmail.com||March 2010. As Mary knows the format of the string used for IBE, she uses the same string to calculate the private key for herself. This approach greatly simplifies the key distribution process and improves security by automatically renewing the cryptographic keys at the end of the specified time period, i.e., at the end of March 2010 in this example. Without IBE, a separate public/private key pair needs to be calculated for every pair of users wishing to communicate with each other. Additionally, every time a new key is generated, it has to be securely distributed throughout the network [Shamir 1985; Boneh and Franklin 2001; 2003].

A number of WSN security issues especially related to key distribution and management can be improved by applying IBE techniques [Shamir 1985; Boneh and Franklin 2001; Cocks 2001]. Several promising BSN research projects, such as [Malan et al. 2004; Lo and Yang 2005; Ganti et al. 2006; Zhong et al. 2006; Patel et al. 2008; Gao et al. 2007], have been making headway. Especially, the possibility of using IBE for BSN [Malasri and Wang 2007; Mont et al. 2003] has been investigated recently. Tan et al. [Tan et al. 2008] developed IBE-Lite to show that an IBE system can be implemented in a resource constrained BSN system. IBE has been applied for delay and disruption tolerant routing [Asokan et al. 2007; Kate et al. 2007] and pervasive computing [Hengartner and Steenkiste 2005] too. However, unlike our approach presented in this paper, data collection and retrieval in these approaches is either limited to the proximity of a BSN or rely on a centralized storage without supporting ubiquitous sensing and monitoring as well as location privacy of patients and load sharing among storage nodes.

3. PROTOCOL FOR SECURE, SCALABLE SENSING AND MONITORING

In this section, our protocol for secure, scalable ubiquitous sensing and monitoring is described.

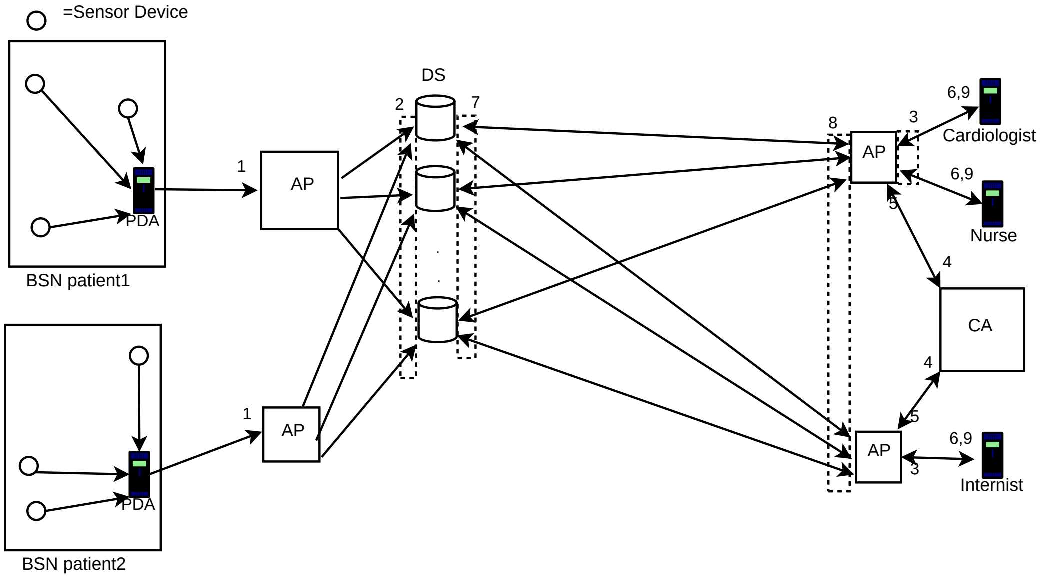

Figure. 1. Overall structure of our security protocol for ubiquitous patient monitoring

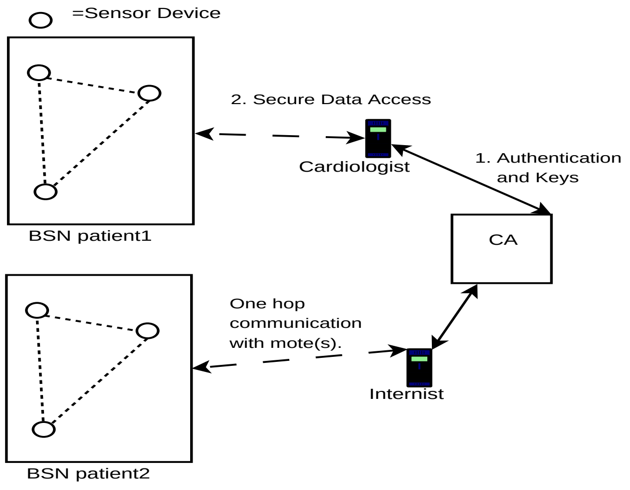

Figure. 2. A sensor network for health care with no storage

3.1 Overview of the Protocol and Assumptions

Figure 1 shows the overall architecture of our protocol for secure anytime/anywhere sensing and monitoring. In addition to supporting direct sensor data collection by a reader in the proximity of a patient, our system supports distributed storage, smart access points (APs) – see Table I, a certificate authority (CA) – see Table I – that supports IBE, and other enhancements to provide secure anytime/anywhere sensing and monitoring, while enhancing the scalability, flexibility, and privacy.

Our architecture shown in Figure 1 contrasts to the architectures adopted in most previous work, in which either the reader and the sensor motes have to always communicate directly as shown in Figure 2 or interact through a centralized storage device as shown in Figure 3. In the first architecture, a reader must be in the proximity to a patient. Although the second approach shown in Figure 3 alleviates this problem, it relies on a centralized storage, which limits the scalability and flexibility.

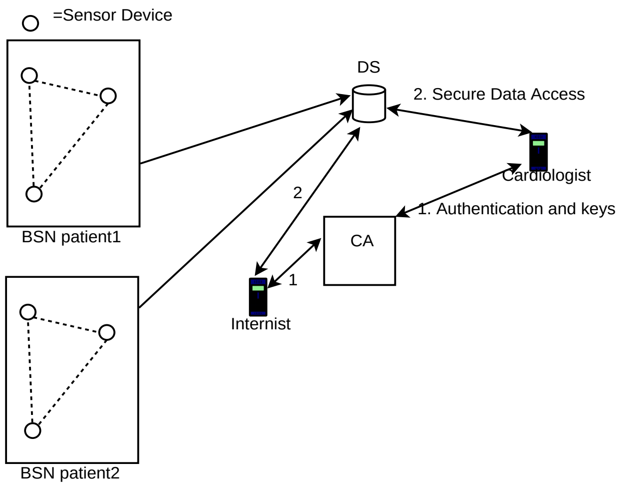

Figure. 3. A sensor network with a central storage

Unfortunately, simply adding more data storage (DS) – see Table I – nodes to the central storage site in Figure 3 without supporting appropriate security mechanisms for anytime/anywhere monitoring of patients cannot solve the scalability and flexibility problem. Instead, it may incur a lot of security and privacy problems concerning sensitive medical data. To avoid these problems, we consider the security and privacy requirements for ubiquitous sensing and monitoring of patients from the very beginning to the end of the protocol design, while addressing the scalability and flexibility limitations of most existing approaches in a secure manner. A description of the **overall procedure** follows:

(1) A patient's PDA (personal digital assistant) – see Table I – or cell phone with an Internet connection collects data from wearable sensors. As shown in Figure 1, it encrypts the data and forwards them to the AP, which subsequently forwards the data to one DS node selected by our scheme for location privacy and load sharing. Notably, the AP and DS in Figure 1 cannot read the encrypted sensor data, because they do not have the private key needed to decrypt the data. Thus, even if an adversary acquires (illegal) access to an AP or DS, he cannot decrypt medical data. In this way, we aim to support fundamental security/privacy requirements associated with individual patients' medical data.

(2) A medical professional's PDA contacts the nearest AP to access patient data. The AP forwards the request to the CA, which verifies whether the professional has the privilege needed to access the requested data. The CA informs the AP of the verification result. If the verification is successful, the CA transmits the encrypted private key to the medical specialist through the AP that does not have the secret needed to decrypt the message and extract the private key.

(3) Also, the AP contacts the specific DS that has the requested data. The DS returns the encrypted data to the medical professional through the AP. Similar to the previous steps, neither the DS nor AP has the private key needed to decrypt the data. Therefore, an adversary cannot read the encrypted medical data, even if he compromises the DS or AP.

As it is very hard, if at all possible, to develop every component for secure information collection and retrieval in a single work, we make the following assumptions:

— We assume that every pair of communication parties, e.g., an AP and DS in Figure 1, authenticates each other (see section 3.2). (Naturally, upon the failure of authentication, a node is disallowed to add data to or read data from any storage.)

— A pair of mutually authenticated nodes can securely exchange temporary shared keys to be used for a specified period of time.

— The CA is well protected and trustworthy.

— All the employed storage sites may store confidential data, but will not maliciously drop or delete the data. This is a reasonable assumption, as third party storage has a financial or legal motivation to be trustworthy.

— In this paper, we do not consider (distributed) denial of service attacks such as the TCP SYN flood attack. Instead, we focus on developing a new protocol to protect the security and privacy of sensitive medical data in a scalable, flexible manner.

To encrypt and decrypt sensitive medical data, we apply the IBE and ECC techniques [Shamir 1985; Hankerson et al. 2004; Menezes et al. 1996]. The authentication of communicating parties mentioned above, the generation of data identifiers for IBE, the ECC scheme underlying IBE in our protocol, and secure data collection, storage, and retrieval procedures are discussed in the next subsections.

3.2 Authentication Key Exchange

Each pair of neighbor nodes in our overlay network undergoes occasional mutual authentication. They then exchange a shared key for use in generating a MAC on each message sent between them until it becomes time to renew authentication of each other. We don't attempt to reinvent the wheel here, instead choosing to use a scheme based on SSL/TLS[Wagner and Schneier 1996; Bhargavan et al. 2008] – see Table I. All parties register with a trusted third party (TTP) – see Table I – such as VeriSign. The TTP issues each node a private key and generates a corresponding public key which is available to any party. This key pair can be generated using any secure scheme, for example VeriSign uses RSA[Rivest et al. 1978] – see Table I.

For any communicating pair, A and B, A can generate a MAC – see Table I – key, M. A then uses B's public key to encrypt M and its own private key to create a digital signature for M. When B receives the signed message, it uses its own private key to decrypt the MAC key, M, and uses A's public key to verify that the message was sent by A and was not tampered with. B can then store the MAC key, and also send back an encrypted and signed message to A, so that A can also authenticate B. Then, the two parties can use the shared key to calculate MACs to communicate with one another until a periodic timeout is reached, at which point they once again undergo mutual authentication.

A → B:
$$[E(MAC_{AB}, PubK_B), DigiSign(E(MAC_{AB}, PubK_B), PrivK_A)]$$

B → A:
$$[E(MAC_{AB}, PubK_A), DigiSign(E(MAC_{AB}, PubK_A), PrivK_B)]$$

This use of asymmetric keys is infrequent compared to our encryption key generation and distribution needs, which are handled by IBE. As mentioned earlier, if we did not use IBE, then we would need to generate separate key pairs for each patient, data type, and reading, and would need to distribute these keys as well. While readings may be quite frequent in the emergency case (multiple per minute) and less frequent in the chronic care monitoring case (hourly), the authentication described above is anticipated to take place daily, if not less frequently.

3.3 Data Identifier

For an IBE system to work, there must be an agreement between patients and readers in regards to the syntax that is used when creating the string used to generate the public key. This string is a **data identifier (ID)** – see Table I – for the information that is being stored or queried. Note that this is the ID of the data and should not be confused with the various patient or reader identifiers used by our system for other purposes. As an example, if there is a patient in this system with patientId = Eric, and his EKG data is required to be stored hourly, a reading taken at 2pm on Sunday, March 15, 2010 is identified by:

ID = Eric:2pm;Monday;15;March;2010:EKG

In this section, we will define several packets labeled (Message 1) − (Message 9) for secure data collection, storage, and access. These messages are numbered consistently in Figure 1 and defined throughout the rest of this section. All of these messages take the general form of:

$$[source||destination||store/load/response||ID||payload]$$

that consists of 1) the source address, 2) the destination address, 3) a store, load, or response command to instruct the receiving node, 4) the data identifier (ID) used for IBE, and 5) the payload, i.e., data. The payload of a packet can include other data, for example, the current location of the patient to send an ambulance to the location in an emergency situation. Note that the entire payload will be encrypted and only the authenticated and authorized medical professionals are allowed to read the payload in our protocol to support the security and privacy of patients.

In the following subsections, we describe the underlying ECC cryptographic primitives, the initialization procedure executed when patients (or their lawful agents) enter the system, the procedure for sensor data encryption and transfer to storage, and the process by which a reader, such as a doctor or nurse, attempts to access data stored in the system.

3.4 Elliptic Curve Cryptography

ECC [Hankerson et al. 2004; Menezes et al. 1996] is a lightweight public key cryptography mechanism. Public key systems generally involve a set of keys, where only the user involved in decryption or digital signing has access to the private key. The public key is freely distributed to any party wishing to encrypt data or verify a signature. Some public key algorithms, such as ECC, require all parties to know a set of predefined constants. However, these domain parameters are not secret, which distinguishes them from the shared secrets which are needed in symmetric key cryptography.

ECC is initialized by selecting a particular elliptic curve defined over a prime field F_p, where p is a large prime number. The curve must adhere to the cubic equation $y^2 = x^3 + ax + b$, where $a, b \in F_p$ are constants. For any chosen elliptic curve, a and b are constants, and all points (x, y) which satisfy the above equation lie on the elliptic curve. The private key is a random number which is multiplied by a generator point, P, which lies on the curve, to derive the public key. P, a, and b are domain parameters of ECC.

The security of ECC is based on the computational difficulty of solving the discrete logarithm problem. Given any two points S and T on the elliptic curve and the knowledge that $S = k \times T$, it is infeasible to derive the scalar k. Therefore, even though the generator point, P, and the public key, Y, are freely available, the private key, X, is secure where $Y = X \times P$. For a more in depth discussion of Elliptic Curve Cryptography, readers are referred to [Hankerson et al. 2004; Menezes et al. 1996].

3.5 Initialization for a New Patient

For each patient that newly enters the system, our system performs the following procedure:

— For each patient, to support **strong security and privacy of sensitive medical data**, the CA chooses a set of n master private keys $(x_1, x_2, ..., x_n)$. After that, it generate a corresponding set of n public keys $(y_1, y_2, ..., y_n)$ instead of using a single public/private key pair. The CA computes $y_i = x_i \times P$ where $1 \leq i \leq n$ using the ECC algorithm. Note that the number n is a tunable parameter with a trade-off between the memory requirement and difficulty of compromising the master private key(s), using a set of temporary IBE-private keys that could be extracted by an adversary, for example, via advanced cryptanalysis. A higher value of n makes a potential compromise harder for higher computational costs and vice versa.

— The master node of the BSN of the patient, e.g., the patient's PDA, are loaded with the master public keys $(y_1, y_2, ..., y_n)$. The CA is loaded with the master private keys $(x_1, x_2, ..., x_n)$.

— Both the CA and patient PDA are loaded with the domain parameters of ECC and a collision-resistant one-way hash function H_{IBE} – see Table I – that hashes a binary input string (a data ID in our system) into an n bit hash value. Using the master public keys and H_{IBE} – see Table I, the master node of the patient's BSN periodically derives a *new temporary* public key to be used for patient monitoring in the specified time interval. The period is predetermined between the CA and patient's PDA. Further, it is equal to the period of medical sensing at which the ID value varies.

— Similarly, the CA periodically re-computes a new temporary private key needed to decrypt the medical data encoded using the temporary public key. From this step, we observe that there is no explicit communication between the BSN master node and CA to set up a pair of public and private keys. Also, the patient's PDA and CA independently renew their temporary public and private keys as the data ID value changes after the specified amount of time.[2]

— Access Points (APs) are loaded with a one-way hash function $H_{storage}$ used to select one of the DS nodes to store the patient's sensor data based on the given data ID, which varies periodically in our protocol as discussed before. Using the same ID, a medical professional authorized (and authenticated) by the CA can request a nearby AP to retrieve the stored data. As the data ID is independent of the geographic location of the patient's, the location of the selected DS node does not expose any additional information about the patient's current/previous location. Also, independent sensor data collection and look-up gives significant flexibility to both patients and medical professionals.

— Any new patient has to be authenticated offline (for example by giving credit card, ssn, insurance information, etc to a sales agent of the company providing this service). At this initial sign up, he can designate his doctors/readers and their permissions. He can then have a secure login/password to contact the CA or some customer service interface which will allow him to add/delete/update readers and permissions. The CA then stores this information in a HashTable to use in later determinations as to when a reader is authorized to access the data. For example, a patient may decide that reader Bob may only access EKG data of patient Alice on Tuesdays. This is then stored by the CA. Later Bob will make a request for a decryption key. If he can prove that he is in fact Bob, and he is requesting EKG readings for Alice from Tuesday, then he will get the key. If either he fails to prove his identity, or he requests something he is not authorized for, then he will be rejected.

3.6 Data Collection and Storage

Data collection is performed by the sensor nodes that are either worn by a patient or installed at the bedside. For the clarity of presentation, let us presume that the patient's PDA (or cell phone), i.e., the master node of the patient's BSN, periodically collects sensor data. In the short term, data may be stored at the master node for direct access during checkups or other direct contact with medical professionals. The patient's PDA then transmits data to distributed storage to enable long-term historical records and allow remote access of the information. When the PDA is ready to transmit data to distributed storage, it first derives the ID of the data, which may take different forms, as long as the syntax is agreed upon ahead of time by both the patient and the reader. In our implementation, we have used the format of:

$$\text{ID} = \text{patientId} + \text{rough time} + \text{data\_type}.$$

For example, Eric:2pm;Tuesday;15;March;2010:EKG is an ID. In this example, the unit of a rough time, which is a tunable parameter, is an hour. Further, EKG is the data type.

Using the ID, the patient's PDA derives an IBE-based public key. It encrypts the data collected in the current time period using the public key and transmits a data store request to the nearest AP as summarized in Algorithm 1. Specifically, the PDA uses the master public key set $(y_1, y_2, ..., y_n)$ and the hash function received from the CA during the initialization to calculate the IBE-public key to be used for this ID:

$$Y_{ID} = \sum_{i=1}^{n} H_{IBE_i}(ID) \times y_i$$

where $H_{IBE_i}(ID)$ is the i^{th} bit of $H_{IBE}(ID)$. As $H_{IBE_i}(ID)$ is either 0 or 1, the result of the above computation is equal to adding a subset of the n master public keys assigned to the patient [Tan et al. 2008]. This subset is determined by the one-way hash of the data ID. The result of the calculation is a new public key, which is the same size as the size of one master public key. Using Y_{ID}, the patient's PDA executes the ECC encryption routine to compute the encrypted sensor data value: E(value, Y_{ID}).

[2]We assume that the master node and CA are time synchronized, for example, using the Network Time Protocol [ntp].

Algorithm 1 BSN $\rightarrow$ AP: The master node of the BSN encrypts data and transmits it to the AP

```
1:  get the public parameters from the CA during the initialization
2:  loop
3:      receive data from sensors periodically
4:      if ready to move to distributed storage then
5:          formulate the ID of the data
6:          derive $Y_{ID}$
7:          $E(value, Y_{ID}) \leftarrow encrypt(data, Y_{ID})$
8:          formulate store request packet (Message 1)
9:          send store request to nearest AP
10:     end if
11: end loop
```

Finally, the patient's PDA sends the store request to its AP. The format of Message 1—the data store request—is as follows:

BSN $\rightarrow$ AP: (Message 1)

$$[BSN\|AP\|store\|ID\|E(value, Y_{ID})\|MAC(K_{BSN-AP}, BSN\|AP\|store\|ID\|E(value, Y_{ID}))]$$

The PDAs and APs are assumed to periodically authenticate each other (see section 3.2). After authenticating each other, the PDA and closest available AP exchange a temporary shared key, K_{BSN-AP}, to compute the message authentication code (MAC) – see Table I, i.e., secure checksum, for each message to be exchanged between them. Using the MAC, the integrity and source of the message can be verified. Specifically, K_{BSN-AP} is used to compute the MAC for $BSN\|AP\|store\|ID\|E(value, Y_{ID})$ in Message 1. Further, the MAC key is periodically renewed and its lifetime is no longer than the authentication period. By integrating mutual node authentication and secure checksums of individual messages, we aim to balance strong node authentication and cost-effective verification of more frequent message transmissions. In the rest of this paper, this approach is consistently applied to every pair of nodes communicating with each other.

To allow patients to have the flexibility to use our system outside of a hospital or even their own smart houses, an AP is simply the first hop on the overlay network using our security protocol for ubiquitous sensing and monitoring. Each BSN or reader is pre-loaded with a list of the IP addresses of potential APs, one of which can be selected based on the connectivity and latency. Since this AP can be at any known location on the Internet, the patient can have as much mobility as their condition allows as long as they have some connectivity, such as WiFi, cellular network, or satellite network to the Internet. As with any overlay network, the usual tunneling mechanisms can be utilized.

Algorithm 2 AP $\rightarrow$ DS: Executed at the AP upon receiving a store request from a patient

```
1: extract ID from the STORE request
2: $index \leftarrow H_{storage}(ID)$
3: $dsLocation \leftarrow routingTable[index]$
4: reformulate store request packet (Message 2)
5: send the store request to the appropriate data store
```

Upon receiving a store request from a patient, the AP executes Algorithm 2. The AP needs to decide where to store the data. Although this can be done in a number of ways, any useful method for determining which DS node to use at any given time should not depend on the geographic location of the patient to support the **location privacy** of the patient, while **distributing load** among storage nodes. Additionally, it should be lightweight to support efficient storage and retrieval of sensor data. To this end, we use a secure, one-way hash function, $H_{storage}$. Using the result of $H_{storage}(ID)$, the AP sends the store request to the selected DS. Hence, the DS is selected independently of the patient's current location. Once a specific storage node is selected via one-way hashing, Message 2 is sent to the selected DS node.

AP → DS: (Message 2)

$$[AP\|DS\|store\|ID\|E(value,Y_{ID})\|MAC(K_{AP-DS},AP\|DS\|store\|ID\|E(value,Y_{ID}))]$$

Due to the nature of one-way hashing, it is very hard for an adversary to reversely derive the patient's location from the location of the selected DS, even if he can find out the selected DS node and its physical location. Also, since different patients have different patient IDs and the data ID value of even the same patient changes periodically, patient data are naturally distributed among DS nodes, enabling load sharing among DS nodes. Using our approach, one can add more DS (and AP) nodes to store and retrieve more data for a potentially increasing number of patients and medical professionals. In general, IBE-based systems are much more efficient than the other public key systems such as RSA; it has been shown to have only the same order of overhead as symmetric key systems [Shamir 1985; Tan et al. 2008]. By applying IBE and ECC in addition to our scalable, distributed protocol design, the **scalability** of secure ubiquitous sensing is substantially improved. Also, the complexity of key management is reduced considerably due to IBE.

The DS node stores the *encrypted data and ID* received from the AP, if the MAC in Message 2 verifies successfully. As discussed earlier, the DS and AP authenticate each other to verify who is intending to write sensor data and whether data is being stored in a legitimate site. Similar to communications between a patient's PDA and AP, there is a periodic authentication system which results in the exchange of a new shared key that is then used to generate and verify a MAC for each message passed between the AP and DS (see section 3.2). Also, note that the AP and DS do not have the private key needed to decrypt the encrypted data. Thus, an adversary cannot decrypt private medical data, even if he/she is able to compromise the AP or DS.

3.7 Retrieving Data

A health care professional needs to be able to read the stored data to fulfill his/her roles. To allow the reader as much flexibility as possible, the AP that the reader contacts is not necessarily be a literal first-hop Internet access point, but merely the known address (or list of addresses to choose from) which the reader needs to communicate with first as its gateway into the system. Thus, the reader can be at any location and receive timely updates as needed. Similar to the previous steps for sensor data storage, a pair communicating nodes periodically authenticates each other and the MAC is computed for every message transmitted between them.

Algorithm 3 Reader → AP: Executed at the PDA or laptop of a medical professional

```
1: $ID \leftarrow patientId : roughtime : data\_type$
2: formulate a LOAD request packet (Message 3)
3: Send (Message 3) to the nearest AP
```

If a reader needs patient information, it executes Algorithm 3. Mutual authentication and shared key exchange for computing the MAC are periodically performed, similar to the date store steps. According to Algorithm 3, the reader sends a load request to an available AP as follows:

Reader → AP: (Message 3)

$$[Reader\|AP\|load\|ID\|readerId\|MAC(K_{R-CA},Reader\|AP\|load\|ID\|readerId)\|$$
$$MAC(K_{R-AP},Reader\|AP\|load\|ID\|readerId\|MAC(K_{R-CA},Reader\|AP\|load\|ID\|readerId))]$$

where the first MAC is computed to support the message integrity between the reader and CA using the shared key, K_{R-CA}, exchanged between them through periodic authentication. The second MAC is computed to support the integrity of the message exchanged between reader and AP using shared key K_{R-AP} that is also periodically renewed.

Upon receiving a load request from a reader, an AP executes Algorithm 4. It starts by extracting the ID from the request packet to calculate $H_{storage}(ID)$ to get the location where the data is stored. The AP verifies the second MAC in Message 3 using K_{R-AP}. If the MAC is successfully verified, the AP replaces it with a new MAC to support the integrity of the message to be forwarded to the CA as follows.

AP → CA: (Message 4)

$$[AP\|CA\|load\|ID\|readerId\|MAC(K_{R-CA},Reader\|AP\|load\|ID\|readerId)\|$$
$$MAC(K_{AP-CA},AP\|CA\|load\|ID\|readerId\|MAC(K_{R-CA},Reader\|AP\|load\|ID\|readerId))]$$

The CA executes Algorithm 5 when it receives a load request from an AP. First, it verifies MAC_{AP-CA}. If the verification succeeds, the CA verifies the MAC_{R-CA} to confirm that the request is in fact from the reader who is identified by readerId and the integrity of the message is maintained. Next, it checks whether or not this reader has the required permission to access the requested data.

If the needed permissions are granted to that readerId, the CA calculates the IBE private key using the master private key set $(x_1,x_2,...,x_n)$, the one way hash function H_{IBE}, and the data ID of the request:

$$X_{ID} = \sum_{i=1}^{n} H_{IBE_i}(ID) \times x_i$$

where $H_{IBE_i}(ID)$ is the i^{th} bit of $H_{IBE}(ID)$. As $H_{IBE_i}(ID)$ is either 0 or 1, the results of the above computation is equal to adding a subset of the n master private keys determined by the one way hash of the data ID. The result of the calculation is a new private key which is the same size as one master public key. The CA encrypts the temporary private key, X_{ID}, and sends it back to the AP in Message 5:

CA → AP: (Message 5)

$$[CA\|AP\|E(response\|ID\|X_{ID},K_{CA-R-E})\|MAC(K_{AP-CA},CA\|AP\|E(response\|ID\|X_{ID},K_{CA-R-E}))]$$

where K_{CA-R-E} is a symmetric key shared between the CA and reader for encryption.

Algorithm 4 AP → CA and DS: Executed at the AP when a LOAD request is received

```
1: From the LOAD request, extract $ID = patientId : roughtime : data\_type$
2: $index \leftarrow H_{storage}(ID)$
3: $dsLocation \leftarrow routingTable[index]$
4: formulate LOAD request packet (Message 4)
5: send LOAD request to CA
6: receive response packet (Message 5) from CA (see algorithm 5)
7: if response is positive then
8:     forward encrypted $X_{ID}$ received from CA to reader (Message 6)
9:     send LOAD request packet (Message 7) to $DS_{index}$
10:    receive response packet (Message 8) from $DS_{index}$
11:    form response packet (Message 9)
12:    send response packet to reader
13: else
14:    send an error message to reader
15: end if
```

If the reader authentication or authorization fails at the CA, the AP returns an error message to the reader. Otherwise, the AP forwards the encrypted private key received from the CA to the reader:

AP to Reader: (Message 6)

$$[AP\|Reader\|E(response\|ID\|X_{ID},K_{CA-R-E})\|$$
$$MAC(K_{AP-R},AP\|Reader\|E(response\|ID\|X_{ID},K_{CA-R-E}))]$$

where K_{CA-R-E} is a symmetric key shared between the reader and CA for encryption, which is renewed as a result of periodic authentication between the reader and CA, and K_{AP-R} is the periodic shared key used to compute the MAC for the messages exchanged between the AP and reader. Note that the AP, any intermediate nodes, or an adversary cannot decrypt the data, because they do not have K_{CA-R-E} needed to decrypt the encrypted private key.

The AP then sends the following load request to the DS node whose address index, DS_{index}, was calculated earlier based on the one-way hash value of the ID:

AP to DS: (Message 7)

$$[AP \| DS_{index} \| load \| ID \| MAC(K_{AP-DS}, AP \| DS_{index} \| load \| ID)]$$

The DS then retrieves the encrypted data which corresponds to ID (see section 4), and returns the following to the AP.

DS to AP: (Message 8)

$$[DS_{index} \| AP \| response \| ID \| E(value, Y_{ID}) \| MAC(K_{DS-AP}, DS_{index} \| AP \| response \| ID \| E(value, Y_{ID}))]$$

where E(value, Y_{ID}) is the encrypted data included in Message 1 and K_{AP-DS} is the MAC key shared between the DS and AP. Upon receiving this response, the AP forwards the encrypted data to the reader:

AP to Reader: (Message 9)

$$[AP \| Reader \| response \| ID \| E(value, Y_{ID}) \| MAC(K_{AP-R}, AP \| Reader \| response \| ID \| E(value, Y_{ID}))]$$

After receiving both the encrypted data and the private key, the reader can decrypt the data.

Overall, to support anytime/anywhere monitoring of patient conditions, our approach supports the security and privacy of sensitive patient data and considerably enhanced flexibility and scalability compared to most existing approaches for medical data collection.

Algorithm 5 Authorize: Executed at the CA when an AP forwards a LOAD request

```
1:  loop
2:    if load request packet (Message 4) is received from an Access Point then
3:      $VERIFY \leftarrow MAC_{R-CA}$
4:      $REQ \leftarrow the\ header\ and\ data\ of\ packet$
5:      if $MAC_{readerID}(REQ) == VERIFY$ then
6:        if hasPermission(readerId, ID) then
7:          calculate $X_{ID}$
8:          formulate response packet (Message 5)
9:          send response packet back to AP
10:       else
11:         send error response to AP
12:       end if
13:     else
14:       send error response to AP
15:     end if
16:   end if
17: end loop
```

4. IMPLEMENTATION AND PERFORMANCE EVALUATION

As discussed before, IBE-based systems are as efficient as symmetric key systems [Shamir 1985; Tan et al. 2008]. To further reduce the overhead, we use shared keys for secure checksums. Also, relatively more expensive parts, such as challenge-response, are not executed as frequently as data transmissions.

Our protocol is written in JAVA using TCP sockets. First, we test the patient/reader client programs on HP iPAQ hx2490b Pocket PCs to ensure that they fall within the RAM, storage, and communications limitations of the embedded device. Each Pocket PC has the Intel 520 MHz XScale processor, 64 MB RAM, 192 MB flash memory, IEEE 802.11b wireless communication, and Microsoft Windows Mobile 5.0 Premium Edition as the operating system.

After that, we run multi-threaded versions of these client programs on Linux desktop machines, to emulate hundreds of concurrent access requests and evaluate the scalability of our system compared to a centralized, IBE-based approach, similar to [Tan et al. 2008]. The Certificate Authority (CA) Access Points (AP), and Data Storage (DS) nodes are implemented as multi-threaded Linux servers. Each Linux machine has the 1.66GHz dual core CPU, 1 GB memory, and 2.6.26 kernel.

One may ask how information search and retrieval can be performed at the appropriate DS, since the data is encrypted. One approach, which is taken by [Tan et al. 2008], is to store each record as a tuple of a flag (commonly known bitstring) and the actual record. Each part of the tuple is encrypted with the same IBE-key. When searching for data, the reader then must read all the encrypted flags sequentially and attempt to decrypt them. If the key he/she has is successful in decrypting the first part of a tuple to produce the known bitstring, then he/she knows to request the second part of that tuple.

This can involve quite a lot of messages and decryptions. Also, sequential search is generally something to be avoided whenever possible. We, instead decided to implement our experiments by storing records as encrypted data indexed by cleartext ID. This is much more efficient, but it it secure? First of all, the DS may not be able to interpret the ID. Second, if the datastore knows the format of the ID, it can figure out that it is storing some data from patientId 13568, from a certain date/time, but still can't decrypt the data itself. Also, knowing a patientId probably won't mean anything to the DS in relation to a real life person. In any case, this is done consistently in both centralized and distributed versions of our experiments, so their comparative performance (see figures 4 through 7) should be unaffected by the decision to use either of these two possible implementations.

We analyze the memory/storage consumption, communication overhead, security, and average service delay as well as scalability in the next subsections.

4.1 Memory and Storage Consumptions

	JVM	Reader	Patient
Memory	14MB	3.99MB	7.78MB
Flash	10MB	6.71KB	13.11KB

Table II. RAM and Flash Memory Consumptions of the JVM, reader client (for a medical professional), and patient client (cluster head)

In each HP pocket PC, 56.59MB of RAM and 133.23MB of flash memory are actually available for running application programs. Table II summarizes the RAM and flash memory consumed by our patient and reader clients. We installed an open-source Java Virtual Machine (JVM)−Mysaifu version 0.4.5 in the pocket PC. The JVM uses about 10MB of flash memory and 14MB of RAM when it is running. The reader client, running on the PDA of a medical professional, uses 3.99MB of RAM in addition to the RAM used by the JVM. The patient client program uses 3.86MB of RAM. The program which allows the patient to authorize new readers and set permissions uses 3.92MB. Although this authorization program can be placed on the same PDA as the patient client program for convenience, it does not necessarily have to be. Also, it is unlikely to run very often. However, to anticipate the worst case, we consider that our protocol uses approximately 21.8MB of RAM on the patient client PDA−two thirds of which are used by the JVM.

In terms of flash memory, the reader client uses 5.55KB for code and an additional 1.16KB to store MAC keys. The patient client program uses 6.11KB and the authorization utility uses another 7KB. These numbers are negligible compared to the amount taken by the JVM itself. The total flash memory usage, including the JVM's usage, is less than 11MB, which is approximately 7.5% of the available flash memory.

4.2 Communication Requirements

Table III. Packet Sizes (Bytes)

Packet Type	Preamble	Payload	Total
Patient Store Request	< 39	64 +20 MAC	< 123
Reader Load Request	< 39	$<$ 10 readerId + 40 MACs	< 89
Response to Reader	< 39	8 + 20 MAC	< 67

In our protocol, the largest packet size is less than 103 bytes. The 802.11 suite allows for up to 2132 bytes in the data portion of its frames, with about 1500 bytes being the typical size utilized by applications to allow for some portion to contain control information from intermediate protocol levels (such as the IP). Thus, we are well within the resource limits of a PDA. Table III shows the packet sizes, in bytes, of those packets sent to or received from a hand-held device, i.e., (Message 1), (Message 3), and (Message 9) described in Section 3. Note that, in our protocol, the single and multiple DS versions use the same size of packets. Thus, distributed data store and load operations do not significantly increase the communication cost.

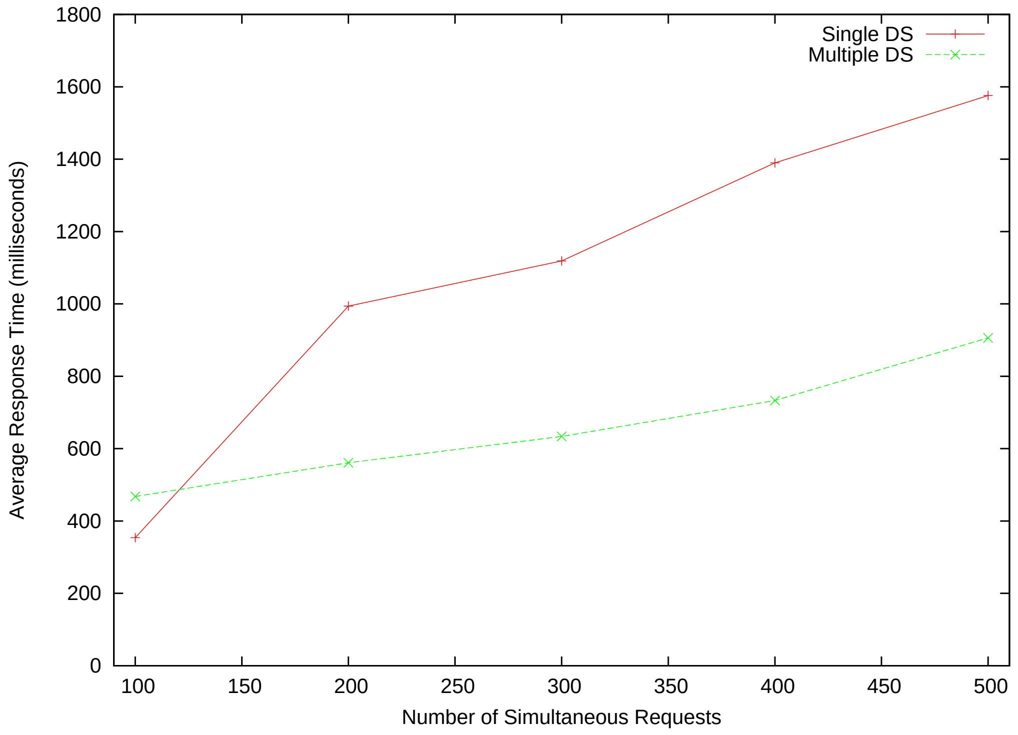

Figure. 4. Response Time for Increasing Store Requests

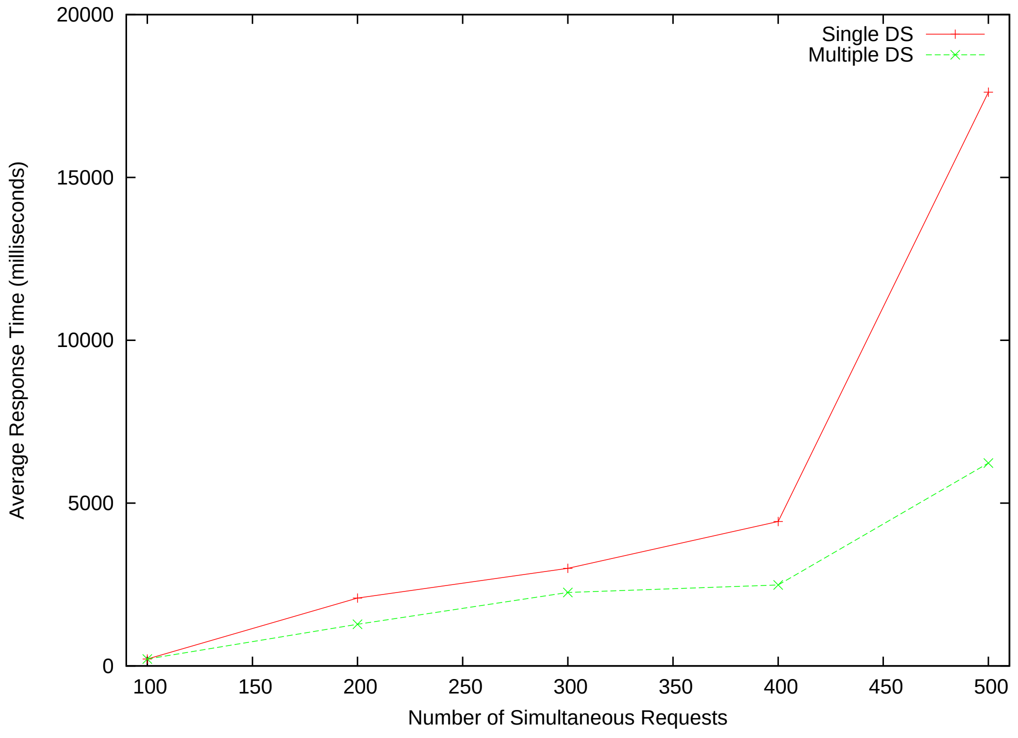

Figure. 5. Response Time for Increasing Load Requests

4.3 Security and Privacy

Our encryption and decryption mechanisms are secure since they are based on IBE [Shamir 1985] and ECC [Hankerson et al. 2004; Menezes et al. 1996] methods – which have already been well established as reliable. Despite the use of X_{ID} and Y_{ID}, instead of simply using an X and Y which are not derived from an ID, the protocol does not violate the discrete logarithm property of Elliptic Curve Cryptography. Since both X_{ID} and Y_{ID} are a straightforward result of the addition of points, the principle that $y = x * P$ is a one way function holds, and thus it is very unlikely that someone could determine X_{ID} given Y_{ID} and P. The use of n secrets keys instead of one, as in the basic ECC setup scheme, ensures that gaining knowledge of one X_{ID} would only allow an adversary to decrypt those messages which have the same ID (or those whose ID happens to hash to the same value). Given a good secure hash function, guessing which IDs would have the the same result would be quite difficult. Even if it could be done, the quantity of data that could be accessed in this fashion would be quite limited.

A captured BSN – meaning, any sensor node or patient PDA – would give nothing to the adversary in terms of retrieving patient data from the storage devices. At most, the adversary would gain only the public parameters of the encryption scheme. Thus, one patient cannot gain access to the data of another, and compromised patient clients would even be thwarted in attempting access of data previously stored by itself.

4.4 Scalability and Performance

In this subsection, we compare the response times of our protocol and a centralized approach using a single, centralized storage device, which represents the current state-of-art in terms of secure patient monitoring [Malasri and Wang 2007; Mont et al. 2003; Tan et al. 2008]. In the first experiment for scalability evaluation, our protocol uses three DS nodes to measure the delay as the number of concurrent service requests increases. Figures 4 and 5 show the delays to process the increasing number of store and read requests, respectively. From the figures, we observe that the average response time of our approach is significantly lower than the response time of the single DS version, which is based on IBE-Lite[Tan et al. 2008]. Also, the difference increases as the number of requests increases. Thus, our approach based on distributed storage is much more scalable than a centralized scheme due to our distributed protocol design for secure ubiquitous monitoring of patients. Although these results are somewhat expected, they are important in that our protocol for secure ubiquitous monitoring advances the state-of-art. It neither constrains patients or medical professionals to a fixed location nor suffers the lack of scalability of centralized approaches to patient data storage and retrieval, while supporting location privacy of patients and load sharing among the storage nodes.

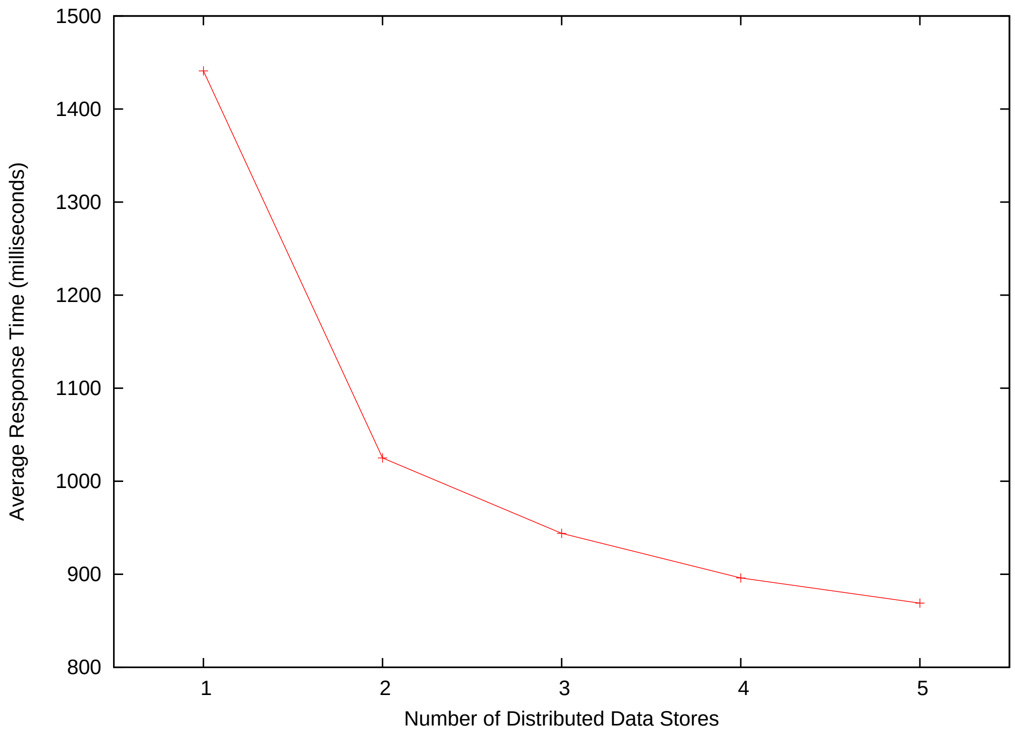

Figure. 6. Response Time for 500 Simultaneous Store Requests

To further analyze the scalability of our approach, we fix the number of simultaneous requests at 500 and then compare the service delays for an increasing number of distributed data stores. These results are shown in Figures 6 and 7. From the figures, we observe that the response time for store and read requests generally decrease as the number of DS nodes increases. Since, our protocol allows for great flexibility in the number of data stores, we have once again demonstrated that our protocol is more scalable than other schemes.

It is notable that, after the system has enough DS nodes to efficiently process data storage and access requests, it reaches a point where adding more DS nodes will have diminishing returns. However, we argue that as the number of concurrent requests increases, the number of DS nodes needed to reach this efficiency point will increase. So, a system administrator who is using our protocol can add more DS nodes, if necessary, to reduce the service delay by observing the relation between the number of DS nodes and service delay by simply adding DS nodes as long as the observed increase in efficiency is substantial.

In our example, 6 or 7 data stores would probably be sufficient, since the marginal benefit of each is decreasing. However, as traffic expands, this flattening of the response curve is likely to be reached at a higher number of data stores. Thus, a system administrator should monitor for an increase in traffic and

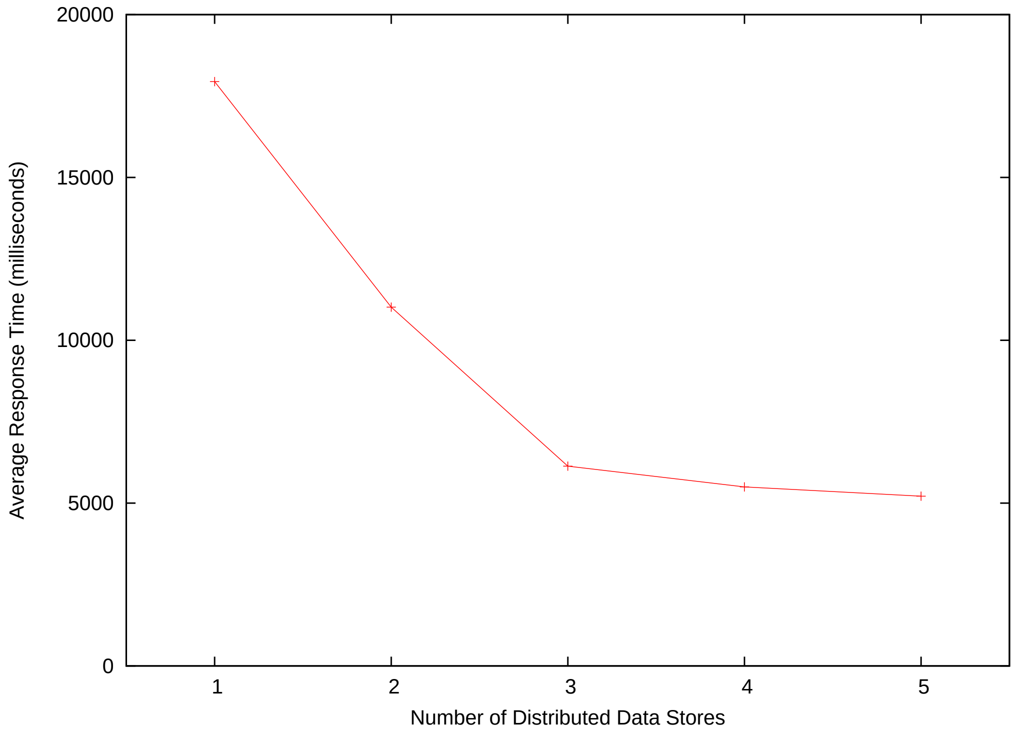

Figure. 7. Response Time for 500 Simultaneous Load Requests

corresponding decrease in response time, then it would be fairly simple to increase the number of data stores. Since a growth in traffic is likely to mean an increase in need for storage capacity anyway, keeping the system at maximum efficiency is simply a matter of adding additional distributed storage sites as storage capacity dwindles instead of replacing existing storage devices with new, higher capacity devices. Doing this in the distributed way is generally a better idea anyway, since it is usually less expensive and will have much less (if any) disruption in service.

This flexibility improves our system's ability to maintain an efficiency level which is much closer to ideal, and continue to keep up with large scale traffic long after a centralized scheme would crash, or be taken down to replace the single DS with a new storage device.

5. CONCLUSIONS AND FUTURE WORK

The cost of caring for elderly and other patients with chronic diseases is ever increasing. Further, confining patients in chronic conditions to a fixed location adversely affects the quality of their lives. Even though security issues in sensor networks have been studied previously, most existing work focuses on monitoring patients in a fixed location such as a hospital or nursing home, often relying on centralized approaches that have relatively poor scalability and flexibility. In this paper, we develop a new protocol to support secure and scalable anytime/anywhere monitoring of patients to allow for more flexibility and efficiency in terms of patient mobility and data access by medical professionals, support the security and privacy of sensitive medical data. Our system design is scalable and easily extendable, if the demand for service increases. In the future, we will investigate more advanced approaches to supporting security and privacy in a cost-effective manner. Additionally, we will explore the possibility of extending our approach to support other applications, such as social networking applications, driving directions, or other mobile applications for which one might like to enhance scalability, lower costs, lower logistical barriers, and allow anytime/anywhere access to data in a secure manner.

REFERENCES

ASOKAN, N., KOSTIAINEN, K., GINZBOORG, P., OTT, J., AND LUO, C. 2007. Applicability of identity-based cryptography for disruption-tolerant networking. In *Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking*.

BHARGAVAN, K., FOURNET, C., CORIN, R., AND ZALINESCU, E. 2008. Cryptographically verified implementations for tls. In *CCS 08: Proceedings of the 15th ACM conference on Computer and communications security*. ACM, New York, NY, USA, 459–468.

BONEH, D. AND FRANKLIN, M. 2003. Identity-based encryption from the weil pairing. *SIAM Journal on Computing 32,* 3.

BONEH, D. AND FRANKLIN, M. K. 2001. Identity-based encryption from the weil pairing. In *Proceedings of the 21st Annual International Conference on Advances in Cryptology*.

cdc-overview 2010. Center for Disease Control and Prevention: Chronic Diseases and Health Promotion. http://www.cdc.gov/chronicdisease/overview/index.htm.

CHAN, H., PERRIG, A., AND SONG, D. 2003. Random Key Predistribution Schemes for Sensor Networks. *IEEE Symposium on Security and Privacy*.

COCKS, C. 2001. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*.

DU, W., WANG, R., AND NING, P. 2005. An efficient scheme for authenticating public keys in sensor networks. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*.

ESCHENAUER, L. AND GLIGOR, V. D. 2002. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*.

GANTI, R. K., JAYACHANDRAN, P., ABDELZAHER, T. F., AND STANKOVIC, J. A. 2006. Satire: a software architecture for smart attire. In *Proceedings of the 4th ACM International Conference on Mobile Systems, Applications and Services*.

GAO, T., MASSEY, T., SARRAFZADEH, M., SELAVO, L., AND WELSH, M. 2007. Participatory user centered design techniques for a large scale ad-hoc health information system. In *Proceedings of the 1st ACM SIGMOBILE International Workshop on Systems and Networking Support for Healthcare and Assisted Living Environments*. 43–48.

GIJSEN, R., HOEYMANS, N., SCHELLEVIS, F., RUWAARD, D., SATARIANO, W., AND VAN DEN BOS, G. 2001. Causes and consequences of comorbidity: A review. *Journal of Clinical Epidemiology 54*, 661–674.

HANKERSON, D., MENEZES, A. J., AND VANSTONE, S. A. 2004. *Guide to Elliptic Curve Cryptography*. Springer.

HENGARTNER, U. AND STEENKISTE, P. 2005. Exploiting hierarchical identity-based encryption for access control to pervasive computing information. In *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*.

KARLOF, C., SASTRY, N., AND WAGNER, D. 2004. TinySec: A link layer security architecture for wireless sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. 162–175.

KATE, A., ZAVERUCHA, G., AND HENGARTNER, U. 2007. Anonymity and security in delay tolerant networks. In *Proceedings of the 3rd International Conference on Security and Privacy for Emerging Areas in Communications*.

LIU, A. AND NING, P. 2008. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*.

LIU, D. AND NING, P. 2003. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM conference on Computer and communications security*. 52–61.

LO, B. AND YANG, G.-Z. 2005. Key technical challenges and current implementations of body sensor networks. In *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks*.

MALAN, D., FULFORD-JONES, T., WELSH, M., AND MOULTON, S. 2004. Codeblue: An ad hoc sensor network infrastructure for emergency medical care. In *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks*.

MALAN, D., WELSH, M., AND SMITH, M. 2004. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In *The First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*.

MALASRI, K. AND WANG, L. 2007. Addressing security in medical sensor networks. In *Proceedings of the 1st ACM SIGMOBILE International Workshop on Systems and Networking Support for Healthcare and Assisted Living Environments*. 7–12.

MENEZES, A. J., VANSTONE, S. A., AND OORSCHOT, P. C. V. 1996. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA.

MONT, M. C., BRAMHALL, P., AND HARRISON, K. 2003. A flexible role-based secure messaging service: Exploiting IBE technology for privacy in health care. In *Proceedings of the 14th International Workshop on Database and Expert Systems Applications*. 432.

ntp. NTP: The Network Time Protocol. http://www.ntp.org/.

PATEL, S., HUGHES, R., HUGGINS, N., STANDAERT, D., GROWDON, J., DY, J., AND BONATO, P. 2008. Using wearable sensors to predict the severity of symptoms and motor complications in late stage parkinson's disease. In *Proceedings of Engineering in Medicine and Biology Society*. 3686–3689.

PATRICK, D. L., DINNE, S., ENGELBERG, R. A., AND PEARLMAN, R. A. 2000. Functional status and perceived quality of life in adults with and without chronic conditions. *Journal Clinical Epidemiology 53*, 779–785.

PERRIG, A., SZEWCZYK, R., WEN, V., CULLER, D., AND TYGAR, J. D. 2001. SPINS: Security protocols for sensor networks. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*.

RIVEST, R. L., SHAMIR, A., AND ADLEMAN, L. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM 21,* 2, 120–126.

SABBAH, E., KANG, K.-D., ABU-GHAZALEH, N., MAJEED, A., AND LIU, K. 2008. An application-driven approach to designing secure wireless sensor networks. *Wireless Communications & Mobile Computing 8,* 3, 369–384.

SHAMIR, A. 1985. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in Cryptology*. 47–53.

STEWART, A., GREENFIELD, S., AND HAYS, R. 1989. Functional status and wellbeing of patients with chronic conditions. *Journal of the American Medical Association 262*, 907–913.

TAN, C. C., WANG, H., ZHONG, S., AND LI, Q. 2008. Body sensor network security: An identity-based cryptography approach. In *Proceedings of the first ACM Conference on Wireless Network Security*.

ČAPKUN, S., BUTTYÁN, L., AND HUBAUX, J.-P. 2003. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing 2,* 1, 52–64.

WAGNER, D. AND SCHNEIER, B. 1996. Analysis of the ssl 3.0 protocol. In *WOEC96: Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce*. USENIX Association, Berkeley, CA, USA, 4–4.

YANG, G.-Z., Ed. 2006. *Body Sensor Networks*. Springer.

ZHONG, L., SINCLAIR, M., AND BITTNER, R. 2006. A phone-centered body sensor network platform: Cost, energy efficiency & user interface. In *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks*. 179–182.

Eric Sabbah is a Assistant Professor in the Department of Computer Systems at the City University of New York, College of Technology. He received his Ph.D. in Computer Science from the State University of New York at Binghamton and M.S. in Computer Science from New York University. His research interests include wireless sensor networks, ubiquitous computing, security and privacy.

Kyoung-Don Kang is an Associate Professor in the Department of Computer Science at the State University of New York at Binghamton. He received his Ph.D. in Computer Science from the University of Virginia in 2003. His research interest includes real-time data services, data intensive computing, wireless sensor networks, and security and privacy.