Verification of Key Establishment Protocols for a Home Health Care System

# Kalvinder Singh [1,2], Vallipuram Muthukkumarasamy [2]
[1] *Australia Development Lab, IBM*
kalsingh@au.ibm.com
[2] *School of Information and Communication Technology, Griffith University*
Gold Coast, v.muthu@griffith.edu.au

Abstract

A Body Sensor Network can be used in a home health care system to monitor the elderly or patients with chronic diseases. The security and requirements of the home health care system is complex. We show how genetic design methodology models the requirements of the health care system. In our system, physiological data can be used to establish keys amongst body sensors, where the sensors have no other prior secret. We show how the requirements of the key establishment protocol can be placed into a Requirement Behaviour Tree. A model is generated from the behaviour tree, and a model checker is used to formally verify the protocol within our system. Implementation of the salient features of each of the protocols is provided. The salient features of the protocols were implemented in TinyOS and run on mica2 motes. The time elapsed, complexity of the code, and memory requirements are analysed in detail.

1. Introduction

The aging population and the increase of chronic diseases have placed an immense financial burden on health services. Body sensors can be used to help reduce their costs. Sensors can be used to remotely monitor elderly patients suffering from chronic diseases and allow them to have relatively independent lives. The uses of body sensor networks are inherently complex. For instance, blood pressure increasing due to exercise is normal. However, blood pressure increasing while at rest could mean a serious medical condition. Sensors may not just measure physiological values, but also body motions, which can lead to a number of different sensors needing to communicate with each other. As the number of heterogeneous sensors increases, so will the complexity of interactions between the sensors.

Figure 1 gives a diagrammatic representation of a proposed home health care system [1]. The diagram shows a patient at home with a number of body sensors that can communicate with a camera sensor, the health controller, and a mobile phone. The cameras may only start recording if the body sensors detect that there may be a medical emergency, such as the patient lying horizontal in the kitchen. Surveillance software, such as S3 [2], can be used to detect if the patient is cleaning the kitchen, or getting something from the ground, or there is actually an emergency. If the software does detect

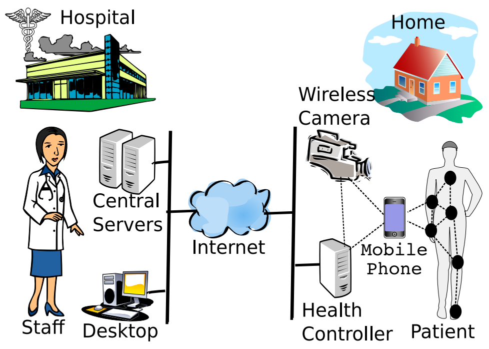

Fig. 1: Home Health Care System

an emergency, the hospital staff is notified; they examine the information, and decide on the best course of action. The mobile phone gives feedback to the patient about the condition of their body, as well as the status of the sensors. The mobile phone can notify the patient of any detected emergency, allowing the patient to report back a false alarm if one has occurred. The mobile phone can be replaced with a PDA or any other hand–held communication device.

Health information collected from sensors needs to be secured and in some countries (for example the USA) security is mandated [3]. Securing a home health care system becomes more difficult mainly because of the different requirements for various components. For instance, the sensors have dramatically more resource constraints than the constraints found in mobile phones, cameras or desktop computers. With differences in computing power, as well as differences in communication costs, different security protocols may be required throughout the entire system. For instance, an efficient key establishment mechanism specifically for body sensors was created using physiological data [1]. However, the home health system may send physiological data to medical staff, or to an analytics engine [4]. The physiological data may also be sent to an actuator to release medicine into the body [4].

When the same physiological data is used for more than one purpose (as well as the complexity of a heterogeneous environment), it becomes important from a security or information assurance point of view to have a formal methodology. A formal methodology is also important to insure that the information sent to medical staff and actuators to dispense medicine is accurate, and the correct actions are taken. The formal methodology has a requirement that it can model both the security and privacy aspects as well as the application correctness. In this paper, we show that Behaviour Trees can be used as a formal methodology to verify both the system and the security.

2. Notation

This paper will use the notations shown in Table 1 to describe security protocols and cryptographic operations.

Table 1: Notations

Notation	Description
A, B	The two nodes who wish to share a new session key
S	A trusted server
N_A, N_B	Random numbers generated by nodes A and B respectively
$[[M]]_K$	Encryption of message M with key K to provide confidentiality
$[M]_K$	One–way transformation of message M with key K to provide integrity
K_{AB}, K'_{AB}	The long–term key initially shared by A and B and the new session key respectively
K_{AS}, K_{BS}	Long–term keys initially shared by A and S, and B and S respectively
X, Y	The concatenation of data strings X and Y
$A \to B : m$	A sends a message m to B
$\xrightarrow{m}$	Another way to define sending of message m
$\oplus$	Exclusive–or function

3. Problems and Limitations

A sensor network can consist of many different computing devices. Some have more computational power (and memory) than others. A Body Sensor Network (BSN) is a network of wearable heterogeneous sensors [5]. The sensors are spread over the entire body, and monitor and communicate a range of health–related data. BSNs are used in the health industry to monitor a patient's physical and biochemical parameters continuously, in different environments and locations wherever the patient needs to go. BSNs can also be used by athletes to measure their performance. Another use for BSNs is controlling characters in video games [5].

Key establishment protocols are used to set up shared secrets between sensor nodes, especially between neighbouring nodes. When using symmetric keys, we can classify the key establishment protocols in Wireless Sensor Networks (WSNs) into three main categories: Pair–wise schemes; Random key predistribution schemes; Key Distribution Center (KDC). The Pair–wise schemes and Random key predistribution schemes are designed for open environments, where there can be many individual sensors [6]. A difficulty with the above schemes is that updating the keys between the nodes is still an unsolved problem. Another drawback is that, when using the random key predistribution schemes, the shared keys cannot be used for entity authentication, since the same keys can be shared by more than a single pair of nodes [7]. The KDC mechanisms by themselves are not suitable for large–scale WSN environments, although combinations of a KDC mechanism and the previously mentioned schemes have created hybrid protocols [8]. Some of the limitations with using a sensor node as a KDC mechanism are:

- The KDC scheme relies upon other schemes to create the trusted intermediary.
- The key sizes in sensor nodes are not large enough, so over a period the key between the sensor and the trusted intermediary may become compromised. If the KDC protocol messages were captured and saved by an adversary, then the adversary may calculate the new keys created.
- Some sensor networks may not need an encryption algorithm, although KDC protocols require an encryption algorithm to encrypt the new key.

A password has been proposed as a way to initiate key establishment [9]. However, the use of a PIN code or a password is not applicable to BSNs since many of the sensors do not have a user-interface. Sensors also may be placed in hard–to–reach places, with some of the sensors implanted into the body. To complicate matters, the sensors may harvest energy directly from the body [10], thus allowing the sensors to exist for long periods. Updating keys is therefore an important function.

This paper uses the generic name *Secure Environmental Value* (SEV) referring to sensed data that can be obtained by sensors in an environment. The SEV is usually hard to obtain through other means. Examples of an environment where SEVs may be found include:

- Human body, where it is difficult to attach a device on the body without the knowledge of the person.
- A secured location, for instance a military base or unmanned vehicle, such as UAVs, or a secure home environment.
- Hard to reach places, for instance a satellite in orbit.

The example environment used in this paper is the human body, where BSNs have been developed to measure the physiological values found in individuals [5]. Health sensors can use Inter–Pulse–Interval (IPI) [11] or Heart Rate Variance (HRV) [12] as good sources for cryptographically random numbers and the physiological values can be used as a one–time pad. Recently, the EKE password protocol was used in BSNs to increase the number of physiological values that can be used [1]. The physiological data replaced the password, in the EKE password protocol. A major limitation to the adoption of the above methodologies is the lack of formal verification.

4. Formal Verification

To our knowledge, there has not been an extensive formal analysis of sensor security protocols. However, formal analysis of communication protocols for traditional networks has been used since at least 1978 [13], with significant improvements in recent decades [14]. Sithirasenan et al. [15] have compared different modeling techniques, and listed advantages for each of the techniques. One of the techniques is the genetic design methodology. One of the major advantages is that the genetic design methodology produces graphical models that are derived and integrated from the original requirements. The models can be used to verify that security protocols correctly work in a complex system.

A home health care system is a complex system, where it is difficult to track how sensed data is used in the system. When the sensed data is also used in security protocols, tracking the use of sensed data becomes even more important. For example, some key establishment protocols require the sensed data never to be sent in the clear or to an untrusted third party, whereas other protocols do not need such restrictions.

The genetic design methodology creates behaviour trees, which in turn can generate SAL code [15]. A model checker can then be used to verify the SAL code and thus verify the protocol in the sensor environment. The main steps with the genetic methodology are: translation of requirements to behaviour trees; integration of behaviour trees; architecture transformation; component behaviour projection; component design. When modelling the entire system, genetic design has significant advantages over UML, state charts or other methods. The advantages include:

- Allows designers to focus on the complexity and design of individual requirements while not having to worry about the detail in other requirements. The requirements can be dealt with one at a time (for both translation and integration).
- The component architecture and the component behaviour designs of the individual components are emergent properties of the design behaviour tree.
- The methodology concentrates on discovery behaviour gaps, which in turn discovers requirement gaps. The focus of direct translation of requirements to design makes it easier to see and find gaps.
- An automated method of mapping changes in requirements to changes in design.

An important part of the genetic design methodology is the behaviour trees. Dormey [16] defined Behaviour Trees as: *a formal, tree–like graphical form that represents behaviour of individual or networks of entities which realize or change states, make decisions, respond–to/cause events, and interact by exchanging information an/or passing control.*

Each requirement can be represented as a behaviour tree; this representation is specifically called a Requirement Behaviour Tree.

5. Key Establishment Protocols

We will investigate two different key establishment protocols: Venkatasubramanian and Gupta[17]; EKE[18]. The Venkatasubramanian and Gupta protocol has a requirement that the sensed data should never be sent in the clear or to an untrusted third party. Whereas the EKE protocol has the requirement that the sensed data should not be sent in the clear or to an untrusted third party, while the protocol has not completed. However, once the protocol is completed the sensed data that the protocol used can be made available.

Venkatasubramanian and Gupta[17] used a single message to send a new key to the neighbouring sensor node, as shown in *Protocol 1*.

Protocol 1 Venkatasubramanian BSN protocol

$$A \rightarrow B : N_A, [N_A]_{RANDKEY}, RANDKEY \oplus SEV$$

The new key $RANDKEY$ is encrypted with the physiological value SEV, which is only known to sensors on a particular person. Sensor node B validates that $RANDKEY$ is correct by verifying the MAC of N_A.

Only cryptographically strong physiological values, such as IPI and HRV, can be used. In addition, modern wireless technology (ultra wideband – UWB, radar) may be used to capture the heart rate remotely. It may encounter security risks when only using IPI and HRV to secure the communication. Other cryptographically weaker physiological values, such as blood pressure, and iron count, are less susceptible to those remote attacks.

The next protocol is the EKE protocol, which is an RSA–based password protocol where the exponent only needs to be 160 bits [18]. The EKE protocol is chosen because other variants of password protocols require exponents of size 1024 bits. The EKE protocol is diagrammatically shown in *Protocol 2*. A drawback of the EKE protocol is that it cannot use ECC [1].

Protocol 2 Diffie–Hellman–based EKE protocol

Shared Information: Generator g of G where $p-1=qr$

$$\begin{array}{lcc}
A & & B \\
r_A \in_R \mathbb{Z}_p & & \\
t_A = g^{r_A} & \xrightarrow{A,[[t_A]]_{SEV_1}} & r_B \in_R \mathbb{Z}_p \\
 & & K_{AB} = t_A^{r_B} \\
 & & t_B = g^{r_B} \\
K_{AB} = t_B^{r_A} & \xleftarrow{[[t_B]]_{SEV_2},[[n_B]]_{K_{AB}}} & \\
 & \xrightarrow{[[n_A,n_B]]_{K_{AB}}} & \text{Verify } n_B \\
\text{Verify } n_A & \xleftarrow{[[n_A]]_{K_{AB}}} &
\end{array}$$

The EKE protocol contains four messages. Node A sends the first message to node B, the message contains the location of A (the location value is in the clear), and the first part of Diffie–Hellman, t_A, is encrypted by the weak key SEV_1. After the first message is sent, node B will calculate the second part of the Diffie–Hellman scheme and hence be able to calculate the session key K_{AB}. Node B then sends the second part of the Diffie–Hellman scheme encrypted by the weak key SEV_2 to node A. The nonce n_B is also sent, encrypted by the session key K_{AB}. The last two messages authenticate both A and B, as well as confirming that they have the session key K_{AB}. The encryption of t_A, t_B, n_A, and n_B can be implemented with an exclusive–or function, as originally described by Bellovin [18].

Depending on which environmental value is measured, and how long the protocol will run, different SEVs may be used for the request and response. However, if the SEV stays constant throughout the running of the protocol, then both SEV_1 and SEV_2 will be the same. The EKE protocol is designed for a constant password throughout the running of the protocol, so similar or same data for both SEV_1 and SEV_2 will not adversely affect the protocol.

The EKE protocol was originally designed to handle small entropy secrets, so that off–line and on–line dictionary attacks are infeasible for an adversary. Another useful feature is that even if the secrets SEV_1 or SEV_2 are compromised or available freely after the running of the key establishment protocol, the session key K_{AB} will remain secure and safe.

Both nonces n_A and n_B are cryptographically strong random numbers, allowing the exclusive–or function to be used for encryption. If any nonce was not cryptographically strong then either $n_A \oplus K_{AB}$ or $n_B \oplus K_{AB}$ operation would allow an adversary to significantly reduce the number of valid K_{AB} values. A characteristic of the EKE protocol is that the nonces are never sent out in the clear, since the nonces are used to encrypt the new key K_{AB}.

The value of p should be chosen wisely [18].The value of p should be as close to $2^N - 1$ as possible for the best security.

A. Modelling

In order to verify the BSN system, the Behaviour Tree technique is used to represent the home health care system. The modelling was completed after several stages. The initial stages involved obtaining the requirements of the Venkatasubramanian and Gupta protocol and EKE password protocol. The major requirement is to establish a cryptographic key between two nodes. The Venkatasubramanian and Gupta protocol properties include that SEV needs to be cryptographically strong, and the SEV should never be sent in the clear. The EKE protocol does not have as many restrictions because of the following properties:

- Sensor nodes only possess a secret of small entropy,
- Off–line dictionary attacks are not feasible,
- On–line dictionary attacks are not feasible, and
- The key must have forward secrecy.

From the properties of the key establishment protocols, we developed the Requirement Behaviour Trees (RBTs). While developing the RBTs, we found that the previous definitions and properties of the protocols did not have a consistent method to define the need for the sensor to sense the physiological data. The RBT is designed for, and has built–in syntax for, external events, so this requirement was easily added to our RBTs. The feature for quickly adding external events makes RBTs suitable for a sensor environment. The RBTs were then placed into an Integrated Behaviour Tree (IBT) to display the entire system. The IBT was then used to create other models for us to investigate and analyse. The Component Interaction Network (CIN) was used to show the relationship between the components in the system and gave a representation of the component architecture. The Component Behaviour Trees (CBTs) and Component Interface Diagrams (CIDs) gave us views of each of the individual components. The final RBT for the EKE protocol is shown in Figure 2. The RBT for the Venkatasubramanian and Gupta protocol has a similar structure.

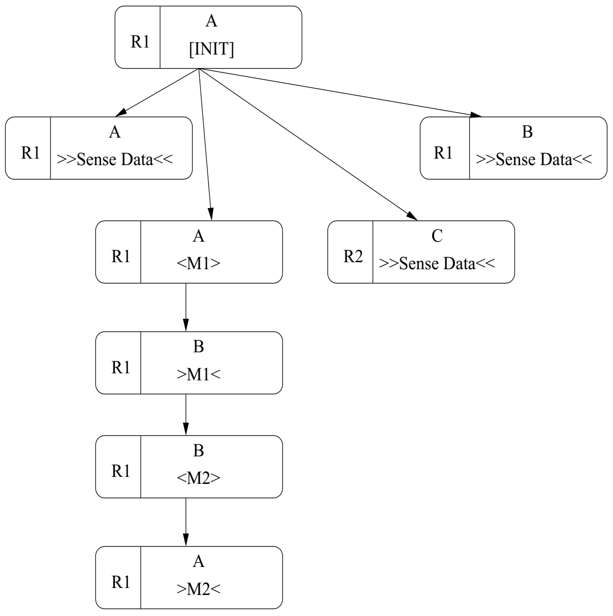

Fig. 2: EKE password protocol for Sensors

The RBT has four major components, the first three components belong to Requirement 1 (R1), whereas Sensor C sensing data belongs to Requirement 2 (R2):

- Sensor A sensing data every 10 seconds
- Sensor B sensing data every 10 seconds
- Sensors A and B Establishing a key
- Sensor C sensing data every 10 seconds

In the above diagram, establishment of the key is initiated by Sensor A. It will create t_A and then send it to Sensor B. In our RBT we have made the sending of the message from Sensor A to Sensor B non–deterministic. In this case, Sensor B could have received a malicious message from another node. Verification of the key is the last step. We have this as a separate RBT, since it overcomplicates the diagram. The verification of the key involves the key confirmation steps described in the protocol.

By using behaviour trees, we were quickly able to find all of the possible inputs and outputs that a sensor can obtain, either through wireless communication or through their sensing devices. This also helps us to verify that each component that we are developing has the needed features to run in our environment. When there are a large number of sensors, this requirement becomes difficult to track. The next step is to generate SAL code from this behaviour trees, and verify the protocol in a sensor environment.

6. Specification of SAL

Before we could test our requirements on the key establishment protocol, we first needed to specify the network and body into SAL code. To specify the network in SAL, we were able to utilize previous SAL libraries [19]. However, we found no existing SAL libraries to specify obtaining SEVs from the body. We defined the body within SAL as having two operations: *getSEV*; *changeSEV*. Sensors can obtain a SEV by calling *getSEV* and afterwards a *changeSEV* can be called to create a new SEV.

We then generated the SAL code from the RBTs. The first SAL code generated is for the Venkatasubramanian and Gupta protocol. Due to limitations in the SAL generation, we modified the SAL code to read the physiological data from our body SAL code. We have a requirement R2 where a sensor sends physiological data to an external third party system. We want to show that requirement R2 will break requirement R1, since for the protocol to be secure we needed to ensure that the sensed data is never sent in the clear. The following theorem is used to verify that no other sensor reads the same sensed data as the pair that is establishing the new session key.

```
prop_no_delay: THEOREM system |-
G(NOT((FORALL (x,y: principals):
(buffer.1.1=buffer.2))));
```

SAL code was also generated for the EKE protocol. We modified the SAL code to read the physiological data from our body SAL code. We have a requirement R2 where a sensor sends physiological data to an external third party system. We want to show that the requirement R2 will not break the requirement R1, since we also placed a delay into the sensors in requirement R1, where the sensor will wait 30 seconds before sending out the physiological data. It should be noted that the Venkatasubramanian and Gupta protocol still is broken if the physiological data is sent out with a delay. The following theorem is used to verify that another sensor delays its send when reading the same sensed data as the pair that is establishing the new session key.

```
prop_delay: THEOREM system |-
G(NOT((FORALL (x,y: principals):
(buffer.1.1=buffer.2 AND
delayed.2=true))));
```

7. Implementation

We implemented and compared different cryptographic primitives that can be used in body sensor security protocols on a Crossbow mica2 MPR2600 mote [20].

Before comparing the different cryptographic primitives, and the benefits that one implementation has over another, we created skeleton code based on TinyOS 2.x [21]. The skeleton code initializes the sensor node, and after the sensor is initialized, we obtained the initial time in milliseconds. We then run a cryptographic primitive in a loop for 2000 iterations, before obtaining a new time. We subtracted the new time from the initial time to obtain the elapsed time in milliseconds to run our cryptographic primitive for 2000 attempts. The elapsed time was then sent via the serial connection, to a PC running a Linux® distribution where we have a Java® application reading the TinyOS packet from the serial port, and report that data to the user.

The key establishment protocols uses exclusive–or (xor) to encrypt the new session key. We compare this method with other methods of encrypting the new session key for body sensor networks. Singh et al. [22], [1] have shown how RC5, SKIPJACK, HMAC–MD5, RSA, and ECC cryptographic primitives can be used in BSNs, however, their work and comparisons were based on simulations, and on TinyOS 1.x. We have implemented these cryptographic primitives on real hardware, and for TinyOS 2.x. To our knowledge these cryptographic primitives have not (until now) been ported to the latest version of TinyOS. Previously, Singh et al. did not separate the square root function from the elliptic curve cryptography. However, in our comparison we found significant information when separating them.

Table 2 shows the time it takes to run 2000 iterations of each of the algorithms. We have ordered the algorithms on the time elapsed. The *Lines of Code* indicates the complexity for the coder to implement the algorithm. The *Size (bytes)* indicates the size in bytes of the application.

Table 2: Time measurements for different algorithms

Algorithm	Time	Lines of Code	Size (bytes)
xor	2 milliseconds	80	6340
RC5	500 milliseconds	506	7168
SKIPJACK	700 milliseconds	697	8138
HMAC–MD5	20 seconds	507	19054
RSA	43 seconds	1456	7814
SQRT	80 seconds	3366	8610
ECC	78 minutes	5038	16328

The RC5 application took considerable more effort than the exclusive–or (xor) application. We found an RC5 implementation for TinyOS 1.x in the TinySEC library [23], however, it has yet to be ported to TinyOS 2.x. Most of our effort was spent porting the code to the new platform.

The SKIPJACK application had similar problems as the RC5 application. Where there was an implementation for TinyOS 1.x in the TinySEC library but there was not one for TinyOS 2.x. Once again, most of our effort was spent porting the code to the platform.

For HMAC–MD5 application we could not find any previous implementations of HMAC–MD5 in any version of TinyOS. In this case we obtained code from RFC1321 [24] and RFC2104 [25] and ported the code to first the nesc language and then to the TinyOS application. This was considerably more effort then either RC5 or SKIPJACK implementations.

The RSA application also had similar problems as the RC5 and SKIPJACK implementations. We found code in the Deluge System [26], however, the RSA code was based off TinyOS 1.x. Effort was required to port this code to TinyOS 2.x. We used a 160 bit exponent as required by the EKE protocol.

The SQRT application had the most difficulties since we implemented it from pseudo–code rather than porting any code. We used Newton's Method [27] for finding square roots to implement the SQRT application.

The ECC application also had similar problems to the RSA, RC5 and SKIPJACK implementations. We ported an ECC library [28] developed for TinyOS 1.x to TinyOS 2.x. The ECC application used a 160 bit points, since password protocols that could be converted to use ECC require stronger keys [1].

The xor application is the quickest by several orders of magnitude compared to the other cryptographic primitives. The size of the application is smaller, and the number of lines is less then the other applications. The xor application is the quickest, whereas the ECC application is the slowest. This verifies existing research into the differences in speed for password protocols of RSA and ECC implementations in TinyOS simulators [1]. The HMAC–MD5 application is the largest, however the application was a straight port from the RFCs, where the code was not intended for sensors.

8. Conclusions and Future Work

Genetic design methodology is used to gather the requirements of the health care system. We examined two existing key establishment protocols that use physiological data to establish keys between body sensors, where the sensors have no other prior secret. We showed how the requirements of the EKE protocol can be placed into a Requirement Behaviour Tree. SAL code is generated from the behaviour tree, as well as SAL code created to model the events from the body. A SAL model checker is used to verify the protocol formally within our system. Implementation of the protocols involved either porting libraries or creating new libraries in TinyOS 2.x. The time elapsed, complexity of the code, and memory requirements are analysed in detail on mica2 sensors. The password protocols that use ECC had a larger computational overhead than the EKE protocol, confirming existing work performed using simulations. Future work will include the full implementation and analysis of both the RBTs and code for each of the key establishment protocols on our sensor network.

Acknowledgments

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. Other company, product, or service names may be trademarks or service marks of others.

References

[1] K. Singh and V. Muthukkumarasamy, “Authenticated key establishment protocols for a home health care system,” in *Proceedings of the Third International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, Melbourne, Australia, December 2007.

[2] A. Hampapur, L. Brown, J. Connell, N. Haas, M. Lu, H. Merkl, S. Pankanti, A. Senior, C.-F. Shu, and Y. Tian, “S3-r1: the ibm smart surveillance system-release 1,” in *ETP ’04: Proceedings of the 2004 ACM SIGMM workshop on Effective telepresence*. New York, NY, USA: ACM Press, 2004, pp. 59–62.

[3] USA, “Summary of hipaa health insurance probability and accountability act,” US Department of Health and Human Service, May 2003.

[4] J. Espina, T. Falck, and O. Mülhens, “Network topologies, communication protocols, and standards,” in *Body Sensor Networks*, G.-Z. Yang, Ed. Springer–Verlag, 2006.

[5] O. Aziz, B. Lo, A. Darzi, and G.-Z. Yang, “Introduction,” in *Body Sensor Networks*, G.-Z. Yang, Ed. Springer–Verlag, 2006.

[6] D. Liu and P. Ning, *Security for Wireless Sensor Networks*, S. Jajodia, Ed. Springer Berlin / Heidelberg, 2007.

[7] P. Hämäläinen, M. Kuorilehto, T. Alho, M. Hännikäinen, and T. D. Hämäläinen, “Security in wireless sensor networks: Considerations and experiments.” in *SAMOS*, 2006, pp. 167–177.

[8] H. Chan and A. Perrig, “PIKE: Peer intermediaries for key establishment in sensor networks,” in *Proceedings of IEEE Infocom*. IEEE Computer Society Press, Mar. 2005.

[9] K. Singh, K. Bhatt, and V. Muthukkumarasamy, “Protecting small keys in authentication protocols for wireless sensor networks,” in *Proceedings of the Australian Telecommunication Networks and Applications Conference*, Melbourne, Australia, December 2006, pp. 31–35.

[10] A. Kansal and M. Srivastava, “Energy–harvesting–aware power management,” in *Wireless Sensor Networks: A Systems Perspective*, N. Bulusu and S. Jha, Eds. Artech House, 2005.

[11] C. C. Y. Poon, Y.-T. Zhang, and S.-D. Bao, “A novel biometrics method to secure wireless body area sensor networks for telemedicine and m–health,” *IEEE Communications Magazine*, vol. 44, pp. 73–81, April 2006.

[12] S.-D. Bao, Y.-T. Zhang, and L.-F. Shen, “Physiological signal based entity authentication for body area sensor networks and mobile healthcare systems,” in *27th Annual International Conference of the Engineering in Medicine and Biology Society, 2005*. IEEE Press, 2005, pp. 2455–2458.

[13] C. West, “General technique for communications protocol validation,” *IBM Journal of Research and Development*, vol. 22, no. 4, 1978.

[14] E. M. Clarke and J. M. Wing, “Formal methods: state of the art and future directions,” *ACM Comput. Surv.*, vol. 28, no. 4, pp. 626–643, 1996.

[15] E. Sithirasenan, S. Zafar, and V. Muthukkumarasamy, “Formal verification of the ieee 802.11i wlan security protocol,” in *Australian Software Engineering Conference (ASWEC ’06)*, Sydney, Australia, 2006.

[16] R. Dromey, “From requirements to design: Formalizing the key steps,” *sefm*, vol. 00, p. 2, 2003.

[17] K. K. Venkatasubramanian and S. K. S. Gupta, “Security for pervasive health monitoring sensor applications,” in *ICISIP ’06: Proceedings of the 4th International Conference on Intelligent Sensing and Information Processing*. Bangalore, India: IEEE Press, December 2006, pp. 197–202.

[18] S. M. Bellovin and M. Merritt, “Encrypted key exchange: Password-based protocols secure against dictionary attacks,” in *IEEE Symposium on Research in Security and Privacy*. IEEE Computer Society Press, 1992, pp. 72–84.

[19] J. Rushby, “The needham-schroeder protocol in sal,” October 2003, http://www.csl.sri.com/users/rushby/abstracts/needham03.

[20] Crossbow, “Crossbow,” http://www.xbow.com/, 2006.

[21] TinyOS, “An operating system for sensor motes,” http://www.tinyos.net/, 2007.

[22] K. Singh and V. Muthukkumarasamy, “Performance analysis of proposed key establishment protocols in multi–tiered sensor networks,” *Journal of Networks*, vol. 3, no. 6, 2008.

[23] C. Karlof, N. Sastry, and D. Wagner, “Tinysec: a link layer security architecture for wireless sensor networks,” in *SenSys ’04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2004, pp. 162–175.

[24] R. Rivest, “The md5 message-digest algorithm,” April 1992, http://tools.ietf.org/html/rfc1321.

[25] H. Krawczyk, M. Bellare, and R. Canetti, “Hmac: Keyed-hashing for message authentication,” February 1997, http://tools.ietf.org/html/rfc2104.

[26] P. K. Dutta, J. W. Hui, D. C. Chu, and D. E. Culler, “Securing the deluge network programming system,” *In the Fifth International Conference on Information Processing in Sensor Networks (IPSN’06)*, April 2006.

[27] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, “Root finding and nonlinear sets of equation,” in *Numerical Recipes: The Art of Scientific Computing*, W. H. Press, Ed. Cambridge University Press, 2007.

[28] A. Liu, P. Kampanakis, and P. Ning, “Tinyecc: Elliptic curve cryptography for sensor networks (version 0.3),” February 2007, http://discovery.csc.ncsu.edu/software/TinyECC/.