15

Information Assurance Protocols for Body Sensors using Physiological Data

Kalvinder Singh[1] and Vallipuram Muthukkumarasamy[2]
[1]*IBM and Griffith University*
[2]*Griffith University*
Australia

1. Introduction

Wireless sensors and actuators have the potential to significantly change the way people live and interact. As the sensors permeate the environment they can monitor objects, space and the interaction of objects within a space. Sensors can monitor a wide range of diverse phenomena by collecting information such as vibrations, temperature, sound, and light. Different sensors have different associated costs. For example, a sensor simply detecting light will have different costs to a sensor recording sound. However, less costly sensors can be used to detect a phenomenon before alerting the more expensive sensors to start their monitoring.

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.

There are many different types of sensor environments, ranging from large areas covered by sensors, to many sensors in a small area (Kuorilehto et al., 2005). Different environments have a wide range of varying characteristics. For instance, sensors placed in large open area are not as physically secure as sensors implanted in an individual's body. The protocols proposed in this paper are mainly designed for our home health care system, although these protocols can also be applied to other environments that have similar security characteristics.

Context awareness is an important aspect of body sensor networks (Thiemjarus & Yang, 2006). For instance, blood pressure increasing due to exercise is normal. But if the blood pressure increases while at rest then that could mean a serious medical condition. Sensors may not just measure the physiological values, but also the body motions, and can lead to a number of different sensors needing to communicate with each other.

Figure 1 gives a diagrammatic representation of our proposed home health care system. The diagram shows a patient at home with a number of body sensors that can communicate with

Source: Biosensors, Book edited by: Pier Andrea Serra,
ISBN 978-953-7619-99-2, pp. 302, February 2010, INTECH, Croatia, downloaded from SCIYO.COM

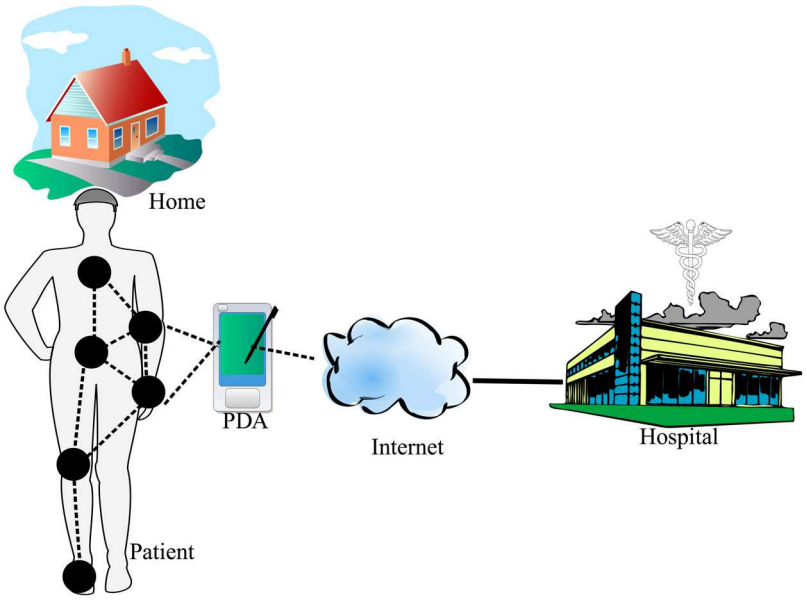

Fig. 1. Architecture and service platform of a BSN for telemedicine and m–health

a camera sensor, the health controller, and a PDA. 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 (Hampapur et al., 2004), can be used to detect if the patient is cleaning the kitchen, or getting something from the ground, or there actually is an emergency. If the software does detect an emergency, the hospital staff are notified, examine the information, and decide on the best course of action. The PDA is used to give feedback to the patient about the condition of their body, as well as the status of the sensors. The PDA can notify the patient of any detected emergency, allowing the patient to report back a false alarm if one has occurred. The PDA can be replaced with a mobile phone or any other hand held communication devices.

Some of the data recorded from body sensors include the heart rate, blood pressure, temperature, and blood oxygen level. They require a data rate of around 2 bits per second (Balomenos, 2001; Yeatman & Mitcheson, 2006). However, other information sent with the message, such as the location of the sender node (8 bits), a MAC (we have specified the size to be the same as the size of the physiological data), and a counter to stop replay attacks (32 bits) raises the data rate to around 10 bits per second. This paper, therefore, assumes the data rate of 10 bits per second for body sensors. Another type of sensor is a surveillance camera, and the data rate requirement for video streams (Axis, 2007) is much greater than that of body sensors. A single camera normally requires 1–4 Mbits per second bandwidth. Providing secure data transfer between sensors is a requirement for our home health care system.

Health information collected from sensors needs to be secured and in some countries (for example the USA) security is mandated (USA, 2003). 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 (Singh & Muthukkumarasamy, 2007). However, the home health system may send physiological data to medical staff, or to an analytics engine (Espina et al., 2006). The physiological data may also be sent to an actuator to release medicine into the body (Espina et al., 2006).

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.

A number of researchers have used environmental data as the only source of secret information to establish keys between body sensors (Bao et al., 2006; Poon et al., 2006; Bao et al., 2005; Bao & Zhang, 2005; Venkatasubramanian & Gupta, 2006). The major benefit of using environmental data is that body sensors can use this information to authenticate that the other sensor is also on the same person and not of another individual. However, these researchers have citepd a number of problems with that approach.

The problems include the following:

- only cryptographically strong environmental data can be used.
- the environmental values can become compromised, in which case the new session key is also compromised.

These problems limit the use of environmental data for establishing keys to only a few cases. The other difficulty our system encounters is in the key establishment scheme of the PDA with sensors at home and in the body. It is envisaged that the patient will simply be users of the system, and will not be able to set up security certificates or keys.

In this paper we propose and develop a number of protocols to address these problems. We show that password protocols can be used to establish keys between body sensors, if passwords are replaced by physiological data. A new protocol is developed to allow a patient to connect a PDA to the home health care system thus able to view information about each of the sensors (ranging from cameras to body sensors). The proposed protocol does not require traditional encryption to transport the new session key. We show that the sensor nodes can establish keys even if no previous shared keys exist between them.

2. Notation

This paper will use the notations shown in Table 2 to describe security protocols and cryptographic operations.

3. Background

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 (Aziz et al., 2006). The sensors are spread over

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
$A \rightarrow B$: m	A sends a message m to B
$\underline{m}$	Another way to define sending of message m
$\oplus$	Exclusive–or function

Table 1. Notations

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 where ever 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 (Aziz et al., 2006). Health information collected from sensors needs to be secured and in some countries, for example the USA, security is mandated (USA, 2003).

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 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 (Liu & Ning, 2007). 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 (Hämäläinen et al., 2006). 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 (Chan & Perrig, 2005). 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 (Singh et al., 2006). 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 (Kansal & Srivastava, 2005), thus allowing the sensors to exist for long periods of time. 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 (Aziz et al., 2006). Health sensors can use Inter–Pulse–Interval (IPI) (Poon et al., 2006) or Heart Rate Variance (HRV) (Bao et al., 2005) as good sources for cryptographically random numbers and the physiological values can be used as a one–time pad. Protocols (Venkatasubramanian & Gupta, 2006; Bao & Zhang, 2005) have been developed that used these physiological values to encrypt a new key between a sensor pair. For instance, Venkatasubramanian and Gupta(Venkatasubramanian & Gupta, 2006) used a single message to send a new key to the neighbouring sensor node, as shown in Protocol 1.

Protocol 1 Venkatasubramanian BSN protocol

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

Venkatasubramanian and Gupta noted that finding additional cryptographically sound physiological values is still an open research problem. Only cryptographically strong physiological values, such as IPI and HRV, can be used. Also, modern wireless technology (ultra wideband – UWB, radar (Staderini, 2002)) may be used to remotely capture the heart rate. 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.

4. Using physiological data to establish keys

Even though PINs and passwords may not be used in body sensors, we show that password protocols can be used. Passwords have low randomness, and therefore have similar characteristics to many SEVs. A four digit PIN contains less than 14 bits of randomness and can be used in a password protocol. A typical password length of eight characters has less than 48 bit of randomness, if we randomly choose upper and lower case letters as well as the digits 0 to 9. We investigate the suitability of password protocols for the sensor environment. Password protocols have the special property of allowing secrets with small entropy to be used for key establishment. Password protocols are designed so that both off–line and on–line attacks are not feasible. A feature or by–product of most password protocols is that if the password is compromised, then any keys created before the password was compromised will not be compromised.
Key sizes in sensor networks are small, normally 64 bits, so that the encryption or integrity tests do not consume a large amount of energy (Karlof et al., 2004). Small key sizes lead to the need to update keys on a regular basis.

Researchers have shown that password protocols can be implemented in sensor networks (Singh et al., 2006). The password protocols were using either a human–entered password on the sensor, or an existing 64 bit key. Instead we propose that environmental data can be used in the absence of using small keys or passwords. Also, previous approaches for sensor environments only used elliptic curve cryptography. However, RSA password protocols that can be converted to ECC have large exponent such as 1024 bits. When the protocol is converted to use ECC, then 160 bit arithmetic is required.

We instead investigated the EKE protocol, which is an RSA based password protocol where the exponent only needs to be 160 bits (Bellovin & Merritt, 1992). 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 (Boyd & Mathuria, 2003).

Protocol 2 Diffie–Hellman–based EKE protocol

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

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

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 & Merritt, 1992).

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 (Bellovin & Merritt, 1992).The value of p should be as close to $2^N - 1$ as possible for the best security.
Even though exclusive–or and block cipher symmetric cryptography is suitable in an RSA environment, it is not suitable when converting to elliptic curves (Boyd & Mathuria, 2003).The EKE (RSA) protocol is compared with a ECC based password protocol (Singh et al., 2006).
Using the RSA implementation (Dutta et al., 2006) from the Deluge system and porting it to the mica2 mote system, and only using 160 bit exponents, we found that the total number of cycles is 147879. Password protocols that can use an ECC implementation inherently require 1024 bit exponents when in RSA mode (Boyd & Mathuria, 2003). When measuring the number of cycles by using the ECC implementation for sensors (Liu et al., 2007), including an implementation of the square root function, we get a total number of 18790689. The key size was 160 bits, which is equivalent to 1024 bits in RSA. When moving to the ECC protocols, more secure keys are required. There is a significant number of extra cycles in a ECC implementation over the RSA implementation.
We also examine the memory of the application, as shown in Table 4. The combination of .bss and .data segments use SRAM, and the combination of .text and .data segments use ROM. The .text contains the machine instructions for the application. The .bss contains uninitialized global or static variables, and the .data section contains the initialized static variables.

Memory	RSA	ECC
ROM	1942	9720
RAM	177	859
.data	60	8
.bss	117	851
.text	1882	9712

Table 2. Memory Overhead In Bytes On MICA2 Platform

We used the values provided by the TOSSIM simulator (a part of the TinyOS installation) to obtain an indication of the power consumption when sending a message. In our calculations we do not take into account any collision avoidance times. On the mica2 mote, the cost of sending an extra 20 bytes is 28.1 microjoules. There is a substantial startup cost for each message sent, and then there is an added cost for every bit that is sent.

4.1 Securely connecting the PDA

When a patient starts up a PDA or obtains a new PDA. Keys need to be created with the PDA and the sensors (both body and camera sensors). The mechanism we use is that the patient, when starting up the PDA, will need to enter either a password or PIN. The remainder of this section discusses a protocol that can be efficiently used to establish keys with all the sensor devices in the household and one of the sensors on the body (normally the body control sensor).

We assume that the body sensors, especially any body sensors that are leader nodes, have obtained the session keys from other sensors in the house. It is envisaged that when the controller sensor was added to the body, it had embedded a key with a central server. Then by using a KDC protocol, it can obtain keys with all the other sensors in the house. When the body sensors establish or update keys between themselves, they can use the password protocols (as described earlier in this paper). However, a hand held device, such as a PDA or mobile phone, may be purchased from a local store and will not have any keys. If patients are required to set up certificates or keys themselves, then the security system may be set up incorrectly. Also, if the device becomes lost or stolen, then an adversary is able to physically obtain any long–term keys held on the device. Our solution is to propose a multiple server protocol that can create session keys between the sensors in the house and the body controller sensor, with the PDA. A multiple server protocol has previously been developed for normal sensor networks (Singh & Muthukkumarasamy, 2006). The main reason for a multiple server protocol is that if sensors exists in an open environment, then KDC nodes can be physically compromised. In our sensor environment, the sensors are less likely to be physically compromised (either they be sensors implanted in the body, or the cameras placed in the home). However, multiple server protocols are still important for a home health care system. The reasons for a multiple server protocol are:

- Increase the randomness of the new key, by having multiple parties adding randomness to the new key.
- A camera may break down, or run out of power. A multiple server protocol increases the availability of the key establishment service.
- The key between the camera and the sensor may become compromised. The keys used by the sensors are normally small in size, since cryptographic algorithms consume more energy when larger keys are used.

In our attempt to create an efficient multiple server protocol, we specified n servers where each server corresponds to a camera. The Proposed Protocol 1 shown below, represents each of the cameras as S_i. The PDA device is labelled as A, and sends the first message, A,B,N_A, to each of the cameras. Each camera sends their message to back to the PDA. The PDA will calculate the keys K_{AS_i} and the cross checksums, and sends the cross checksums as well as parts of the messages from the cameras to the body sensor. The body sensor creates its own cross–checksums and compares them against the cross– checksums created by the PDA. At this stage, the keys K_S and K_{AB} are created by the body sensor. The body sensor sends N_B, the keying data, and the its newly created cross–checksums to the PDA. The PDA can now also create the keys K_S and K_{AB}. The final message completes the key confirmation between the PDA and the body sensor, as shown in Proposed Protocol 1. If key confirmation is not vital, then the final message can be removed.

The Proposed Protocol 1 provides key authentication, key freshness and key confirmation, using multiple authentication servers. In our Proposed Protocol 1, the following constructs are used: π is the password or SEV, A is the PDA, B is a body sensor, S_i is a camera, t_{AS_i} and t_{S_iA} are the Diffie–Hellman values, $m_{AS_i} = [[t_{AS_i}]]_\pi$, $m'_{AS_i} = [[t_{S_iA}]]_\pi$, $AUTH_{Ai} = [A,B,K_i]_{K_AS_i}$, $MASK_{Ai} = [[AUTH_{Ai}]]_{K_{AS_i}}$, $AUTH_{Bi} = [A,B,K_i]_{KBS_i}$, and $MASK_{Bi} = [[AUTH_{Ai}]]_{K_{BS_i}}$

The PDA, the body sensor and the cameras contribute to the key value. The values N_A and N_B are generated by the PDA and the body sensor respectively as input to the MAC function, that determines the session key. The key used with the MAC function is generated by the servers. Both the PDA and body sensor compute the session key as $K_{AB} = [N_A,N_B]_{K_S}$.

Proposed Protocol 1 A Preliminary Multiple Server Protocol

$M1$	$A \rightarrow S_i:$	m_{AS_i}, A, B
$M2$	$S_i \rightarrow A:$	$m'_{AS_i}, S_i, AUTH_{Ai}, MASK_{Ai} \oplus K_i,$ $MASK_{Bi}, AUTH_{Bi} \oplus K_i$
$M3$	$A \rightarrow B:$	$S_1, MASK_{B1}, AUTH_{B1} \oplus K_1, \ldots,$ $S_n, MASK_{Bn}, AUTH_{Bn} \oplus K_n,$ $cc_A(1), \ldots, cc_A(n), N_A, A$
$M4$	$B \rightarrow A:$	$cc_B(1), \ldots, cc_B(n), N_B, [N_A]_{K_{AB}}$
$M5$	$A \rightarrow B:$	$[N_B]_{K_{AB}}$

The keys K_{AS_i} are generated by computing the diffie–hellman part of the protocol. The PDA and body sensor should have a minimum number of cameras returning valid results before confirming that the key is valid. The PDA will calculate $cc_A(i)$ $\forall i \in 1, \ldots, n$.

$$cc_A(i) = \begin{cases} [K_i]_{K_i} & \text{if valid,} \\ EM & \text{otherwise} \end{cases} \tag{1}$$

Where EM is an error message; an example will be the value zero. There is a remote chance a valid case may be zero. If the valid value is zero, the camera needs to be considered a compromised server (even though it is not a malicious server).
The body sensor will calculate $cc_B(i)$, and compare it with $cc_A(i)$. If they are the same, then the server S_i is valid. Below is a way the PDA and body sensor compare the cross checksum for $cc_A(i)$ and $cc_B(i)$.

$$cc_B(i) = \begin{cases} cc_A(i) = [K_i]_{K_i} & \text{if valid,} \\ EM & \text{otherwise} \end{cases} \tag{2}$$

After the comparison of the entire cross checksums, a set of valid keys $V_1, \ldots, V_m$ should remain. The creation of K_S is defined as follows.

$$K_S = V_1 \oplus \ldots \oplus V_m \tag{3}$$

Where V_i is the i^{th} valid key given by a server, and m is the total number of valid servers $t \leq m \leq n$, where t is the minimal number of trusted servers. Another advantage of the proposed protocol is that the cameras will not be able to calculate K_S. The calculated $cc_B(i)$ values are returned to the PDA, where the PDA performs similar checks as the body sensor and calculates K_S.
Once the PDA has established a key with one of the body sensors, then a KDC protocol can be used to establish keys with the other body sensors.

4.2 Analysis and discussion

Our Proposed Protocol 1 has a number of advantages, one of which is that the body sensor does not need good random number generators to create the nonces. The body sensor could even safely use a counter for their nonce values. Another advantage is that if a camera or a number of cameras are unavailable, the authentication service itself still exists through the working cameras. If one or more cameras become compromised, the authentication service or the security of the system is not compromised.

The proposed protocol only encrypts random information. If the encryption cipher uses an *IV* value (such as RC5 and SKIPJACK currently used in TinyOS (TinyOS, 2007)) then we can use a constant *IV* value. However, the constant *IV* value chosen for our protocol must only be used to encrypt the random data and should never be used to encrypt other information. Also, a wide variation of different ciphers can safely be used.

Some MACs have vulnerabilities when the message sizes are variable. All of our message sizes are of constant value, allowing us to safely use a wider range of MACs than previously available. The size of the MACs sent to the body sensor can be lower than that of conventional protocols. The integrity checking is performed by the body sensor. If x is the size of the MAC in bits, then an adversary has 1 in 2^x chance of blindly forging a valid MAC for a particular message. The adversary should be able to succeed in 2^{x-1} tries. Because of the low bandwidth of sensor nodes, a 4 byte MAC, requiring 2^{31} packets, will take years to complete. If an adversary did attempt this attack, the sensor node would be non–functional within that period. In addition, an adversary will need to forge $2t$ MACs; t MACs to A and t MACs to B, and stop traffic from the other base stations before they can determine the value of K_{AB}.

In the proposed protocol, the device that is most sensitive to energy restrictions is the body controller sensor. The message $M3$ is of the most concern, since it the largest message sent to the controller sensor. We calculate the size of the message as $M3 = (n+1)a_0 + a_1 + na_2 + na_3$ bytes. Where a_0 is the size of the location, a_1 is the nonce size, a_2 is the key size, a_3 is the MAC size, and n is the number of cameras. Assuming that the location is 1 byte in size (maximum 256 possible sensors), the nonce is 1 byte in size, the key is 8 bytes in size, and the MAC is 4 bytes in size, we get $M3 = 13n + 2$ bytes. If we assume that a packet size is 28 bytes, a configuration with more than two cameras will require multiple packets sent between the PDA and the body controller node. If there is no or little concern about whether the cameras or the camera keys are compromised, then the PDA can select two cameras to send to the body controller sensor.

The computational complexity for the body sensor depends on the number of valid servers the PDA forwards to the sensor, the number is defined as m. The computational cost of the MACs is $4m + 2$, and the cost of the encryption operations is m. The number of exclusive–or operations is $2m$.

5. Formal verification

Formal analysis of communication protocols for traditional networks has been used since at least 1978 (West, 1978), with significant improvements in recent decades (Clarke & Wing, 1996). Sithirasenan et al. (Sithirasenan et al., 2006) have compared different modeling techniques, and listed advantages for each of the techniques.

Verifying a protocol is proving that the claims for the protocol are correct and is a significant step in analysing the protocol. The complexity of security protocols makes their verification a difficult task. Informal arguments about protocol correctness are not reliable or acceptable, leading to a formal analysis to verify that a claim made by a protocol is correct.

Computer assisted formal methods for verifying security protocols can be divided into two major categories:

- Model Checking: considers a finite number of possible protocol behaviours and allows checking that satisfy a set of correctness conditions. This method works well for finding

attacks on a protocol, rather than proving their correctness Clarke et al. (2000); Lowe (1996); Mitchell et al. (1997).

- Theorem Proving: considers all possible protocol behaviours, and checks that they satisfy a set of correctness conditions. This method works well for proving protocol correctness, rather than finding attacks on protocols Meadows (1996); Paulson (1998); Song (1999).

Both model checking and theorem proving methods require computer assistance to aid with the analysis. However, methods based on theorem proving are less automated than those based on model checking.

A useful feature of model checking methods is that they can prove an attack when a protocol is found not to satisfy a correctness condition. The failure to find an attack indicates that the protocol is correct. However, model checkers do not provide a symbolic proof that can explain why a protocol is correct and thus are uninformative when checking a correct protocol. Another important limitation of model checking methods is that they only guarantee correctness of a scaled down version of the protocol.

Theorem proving mechanisms have their own strengths and limitations. One of the strengths of theorem proving methods is that they can provide a symbolic proof when a protocol is found to be correct. Their main limitation is that they generally require more expert human guidance than methods based on model checking.

Sitherasenan et al. Sithirasenan et al. (2006) have used Genetic Design Methodology to check the correctness of the 802.11i wireless security protocol. The requirements of the protocol was placed into a number of Requirement Behaviour Trees. The requirements were then verified by integrating them into a single Integrated Behaviour Tree. Thereafter, the Behaviour Tree model was translated into SAL formal notations for theorem proving. This mechanism shows that both model checking and theorem proving can be performed using the same analysis tool. However, the model checking was mainly focused on the protocol correctness and not the security. We will show that this analytical tool can perform both model checking and theorem proving on the security of a protocol.

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 (Sithirasenan et al., 2006). 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 (Dromey, 2003) 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.

Another mechanism to verify that a protocol is secure is to use a mathematical proof Canetti & Krawczyk (2001). Problems with using mathematical proofs include:

- With each small change in the protocol a new proof needs to be constructed.
- Security proofs run to several pages of mathematical reasoning and is difficult to understand to the average practitioner.
- There are relatively few protocols with mathematical security proofs.
- As a system becomes more complex, constructing mathematical proofs becomes more challenging.

Informal verification, machine analysis (either using model checking, or theorem proving), and mathematical proofs are all important approaches to gain assurance on the security of the protocol.

5.1 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.

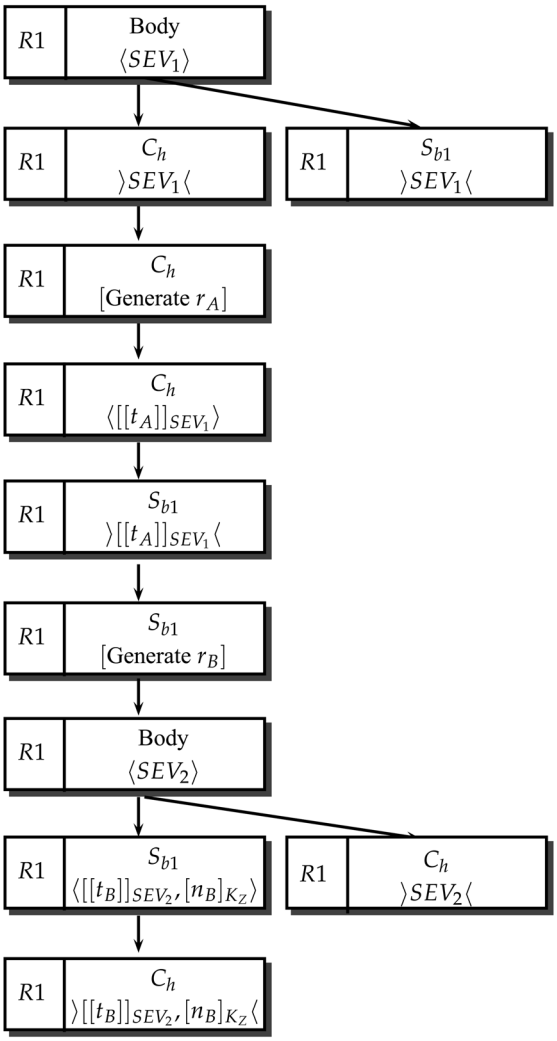

Fig. 2. EKE password protocol for Sensors

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.

5.2 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 (Rushby, 2003). 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. Wemodified 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))));
```

6. Comparison of different implementation

We implemented and compared different cryptographic primitives that can be used in body sensor security protocols on a Crossbow mica2 MPR2600 mote (Crossbow, 2006). Before comparing the different cryptographic primitives, and the benefits that one implementation has over another, we created skeleton code based on TinyOS 2.x (TinyOS, 2007). 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. (Singh & Muthukkumarasamy, 2008; 2007) 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 6 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.

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

Table 3. Time measurements for different algorithms

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 (Karlof et al., 2004), 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 (Rivest, 1992) and RFC2104 (Krawczyk et al., 1997) 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 (Dutta et al., 2006), 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 (Press et al., 2007) 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 (Liu et al., 2007) 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 (Singh & Muthukkumarasamy, 2007).

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 (Singh & Muthukkumarasamy, 2007). 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.

7. Future research directions

We have proposed a multi–server key establishment protocol that allows a PDA to obtain session keys with most of the sensors in our home health care system. We implemented salient features of the password protocols and compared the energy consumption of the nodes. The password protocols that could be converted to use ECC had a larger computational overhead than the EKE protocol, because of the stronger keys required by the ECC–based password protocols. Due to the EKE protocol only requiring 160 bit exponents, the message sizes of the EKE protocol were comparable to the ECC– based password protocols. The impact on memory by adding elliptic curves to a sensor application was analyzed, revealing that there is additional costs associated with an ECC solution over a RSA solution. Future work includes using cryptographic protocol verifiers to confirm that our protocols are secure.

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.

8. 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.

9. References

Axis (2007). Setting up an ip–surveillance system using axis cameras and axis camera station software, http://www.axis.com/files/manuals/gd_ipsurv_design_en_070320.pdf.

Aziz, O., Lo, B., Darzi, A. & Yang, G.-Z. (2006). Introduction, *in* G.-Z. Yang (ed.), *Body Sensor Networks*, Springer–Verlag.

Balomenos, T. (2001). User requirements analysis and spcification of health status analysis and hazard avoidance artefacts, *Technical report*, DC FET Project ORESTELA, Delieverable D02.

Bao, S.-D. & Zhang, Y.-T. (2005). A new symmetric cryptosystem of body area sensor networks for telemedicine, *6th Asian–Pacific Conference on Medical and Biological Engineering*. http://ifmbe-news.iee.org/ifmbe-news/july2005/shudibaopaper.html.

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

Bao, S.-D., Zhang, Y.-T. & Shen, L.-F. (2006). A design proposal of security architecture for medical body sensor networks, *BSN '06: Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*, IEEE Computer Society,Washington, DC, USA, pp. 84–90.

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

Boyd, C. & Mathuria, A. (2003). *Protocols for Authentication and Key Establishment*, Springer Berlin / Heidelberg.

Canetti, R. & Krawczyk, H. (2001). Analysis of key-exchange protocols and their use for building secure channels, *EUROCRYPT 2001: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, Springer-Verlag, London, UK, pp. 453– 474.

Chan, H. & Perrig, A. (2005). PIKE: Peer intermediaries for key establishment in sensor networks, *Proceedings of IEEE Infocom*, IEEE Computer Society Press.

Clarke, E. M., Jha, S. & Marrero, W. (2000). Verifying security protocols with brutus, *ACM Transactions Software Engineering Methodology* 9(4): 443–487.

Clarke, E. M. & Wing, J. M. (1996). Formal methods: state of the art and future directions, *ACM Comput. Surv.* 28(4): 626–643.

Crossbow (2006). Crossbow, http://www.xbow.com/.

Dromey, R. (2003). From requirements to design: Formalizing the key steps, *sefm* 00: 2.

Dutta, P. K., Hui, J. W., Chu, D. C. & Culler, D. E. (2006). Securing the deluge network programming system, *In the Fifth International Conference on Information Processing in Sensor Networks (IPSN'06)*.

Espina, J., Falck, T. & Mülhens, O. (2006). Network topologies, communication protocols, and standards, *in* G.-Z. Yang (ed.), *Body Sensor Networks*, Springer–Verlag.

Hämäläinen, P., Kuorilehto, M., Alho, T., H¨annik¨ainen, M. & H¨am¨al¨ainen, T. D. (2006). Security in wireless sensor networks: Considerations and experiments., *SAMOS*, pp. 167–177.

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

Kansal, A. & Srivastava, M. (2005). Energy–harvesting–aware power management, *in* N. Bulusu & S. Jha (eds), *Wireless Sensor Networks: A Systems Perspective*, Artech House.

Karlof, C., Sastry, N. & Wagner, D. (2004). Tinysec: a link layer security architecture for wireless sensor networks, *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, ACM Press, New York, NY, USA, pp. 162–175.

Krawczyk, H., Bellare, M. & Canetti, R. (1997). Hmac: Keyed-hashing for message authentication. http://tools.ietf.org/html/rfc2104.

Kuorilehto, M., H¨annik¨ainen, M. & Hämäläinen, T. D. (2005). A survey of application distribution in wireless sensor networks, *EURASIP J. Wirel. Commun. Netw.* 5(5): 774–788.

Liu, A., Kampanakis, P. & Ning, P. (2007). Tinyecc: Elliptic curve cryptography for sensor networks (version 0.3). http://discovery.csc.ncsu.edu/software/TinyECC/.

Liu, D. & Ning, P. (2007). *Security for Wireless Sensor Networks*, Springer Berlin / Heidelberg.

Lowe, G. (1996). Breaking and fixing the needham-schroeder public-key protocol using fdr, *TACAs '96: Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, Springer-Verlag, London, UK, pp. 147–166.

Meadows, C. A. (1996). The nrl protocol analyzer: An overview, *Journal of Logic Programming* 26: 113–131.

Mitchell, J. C., Mitchell, M. & Stern, U. (1997). Automated analysis of cryptographic protocols using mur/spl phi/, *SP '97: Proceedings of the 1997 IEEE Symposium on Security and Privacy*, IEEE Computer Society, Washington, DC, USA, p. 141.

Paulson, L. C. (1998). The inductive approach to verifying cryptographic protocols, *Journal of Computer Security* 6(1-2): 85–128.

Poon, C. C. Y., Zhang, Y.-T. & Bao, S.-D. (2006). A novel biometrics method to secure wireless body area sensor networks for telemedicine and m–health, *IEEE Communications Magazine* 44: 73–81.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. P. (2007). Root finding and nonlinear sets of equation, *in* W. H. Press (ed.), *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press.

Rivest, R. (1992). Themd5 message-digest algorithm. http://tools.ietf.org/html/rfc1321.

Rushby, J. (2003). The needham-schroeder protocol in sal. http://www.csl.sri.com/users /rushby/ abstracts/needham03.

Singh, K., Bhatt, K.&Muthukkumarasamy, V. (2006). Protecting small keys in authentication protocols for wireless sensor networks, *Proceedings of the Australian Telecommunication Networks and Applications Conference*, Melbourne, Australia, pp. 31–35.

Singh, K. & Muthukkumarasamy, V. (2006). A minimal protocol for authenticated key distribution in wireless sensor networks, *ICISIP '06: Proceedings of the 4th International Conference on Intelligent Sensing and Information Processing*, IEEE Press, Bangalore, India, pp. 78–83.

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

Singh, K. & Muthukkumarasamy, V. (2008). Performance analysis of proposed key establishment protocols in multi–tiered sensor networks, *Journal of Networks* 3(6).

Sithirasenan, E., Zafar, S. & Muthukkumarasamy, V. (2006). Formal verification of the ieee 802.11i wlan security protocol, *Australian Software Engineering Conference (ASWEC '06)*, Sydney, Australia.

Song, D. X. (1999). Athena: a new efficient automatic checker for security protocol analysis, *CSFW '99: Proceedings of the 12th IEEE workshop on Computer Security Foundations*, IEEE Computer Society, Washington, DC, USA, p. 192.

Staderini, E. M. (2002). Uwb radars in medicine, *IEEE Aerospace and Electronic Systems Magazine* 21: 13–18.

Thiemjarus, S. & Yang, G.-Z. (2006). Context–aware sensing, *in* G.-Z. Yang (ed.), *Body Sensor Networks*, Springer–Verlag.

TinyOS (2007). An operating system for sensor motes, http://www.tinyos.net/.

USA (2003). Summary of hipaa health insurance probability and accountability act, US Department of Health and Human Service.

Venkatasubramanian, K. K. & Gupta, S. K. S. (2006). Security for pervasive health monitoring sensor applications, *ICISIP '06: Proceedings of the 4th International Conference on Intelligent Sensing and Information Processing*, IEEE Press, Bangalore, India, pp. 197–202.

West, C. (1978). General technique for communications protocol validation, *IBM Journal of Research and Development* 22(4).

Yeatman, E. & Mitcheson, P. (2006). Energy scavenging, *in* G.-Z. Yang (ed.), *Body Sensor Networks*, Springer–Verlag.

Biosensors

Edited by Pier Andrea Serra

ISBN 978-953-7619-99-2
Hard cover, 302 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

A biosensor is defined as a detecting device that combines a transducer with a biologically sensitive and selective component. When a specific target molecule interacts with the biological component, a signal is produced, at transducer level, proportional to the concentration of the substance. Therefore biosensors can measure compounds present in the environment, chemical processes, food and human body at low cost if compared with traditional analytical techniques. Bringing together researchers from 11 different countries, this book covers a wide range of aspects and issues related to biosensor technology, such as biosensor applications in the fields of drug discovery, diagnostics and bacteria detection, optical biosensors, biotelemetry and algorithms applied to biosensing.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821