Using Physiological Signals for Authentication in a Group Key Agreement Protocol

Kalvinder Singh
IBM, Australia Development Lab
and Griffith University
St Leonards, Australia, NSW
Email: kalsingh@au.ibm.com

Vallipuram Muthukkumarasamy
Griffith University
Queensland, Australia
Email: v.muthu@griffith.edu.au

***Abstract*—A Body Area Network (BAN) can be used to monitor the elderly people or patients with chronic diseases. Securing broadcasted data and commands within BANs is essential for preserving the privacy of health data and for ensuring the safety of the patient. We show how a group key can be securely established between the different sensors within a BAN. The proposed mechanism uses the inherent secure environmental values. An implementation of the protocols is carried out on mica2 motes and performance is examined in detail. The time elapsed, complexity of the code and memory requirements are analysed. The results confirm the potential benefits in real-world application. We show that a key establishment protocol based on RSA has advantages over a protocol based on ECC for this application.**

I. Introduction

The aging population and the increase of chronic diseases have placed an immense financial burden on health services. A home health care system, with both body and external sensors may be used to help reduce the costs. Sensors can be used to remotely monitor elderly patients suffering from chronic diseases and allow them to have relatively independent lives. Home health care systems require a degree of real–time data collection. A patient with a medical emergency will require data sent to medical staff as quickly as possible, rather than the data sent after a few hours or days. The problem gets even more complex with the requirement that commands sent to sensors need to be in real–time. For instance, if a device recognizes a possible medical emergency, it should notify other sensors immediately to start recording.

Security or information assurance is a very important criteria of the home health care system. A device broadcasting a command to all the sensors should have the transmission secured using a group key. The group key should be shared amongst the device and the sensors.

A home health care system [1], with both body and external sensors is an example of a complex sensor network system. The complexity of the system increases as we add more sensors to collect more data. For instance, blood pressure increasing due to exercise is normal. However, increase in blood pressure 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.

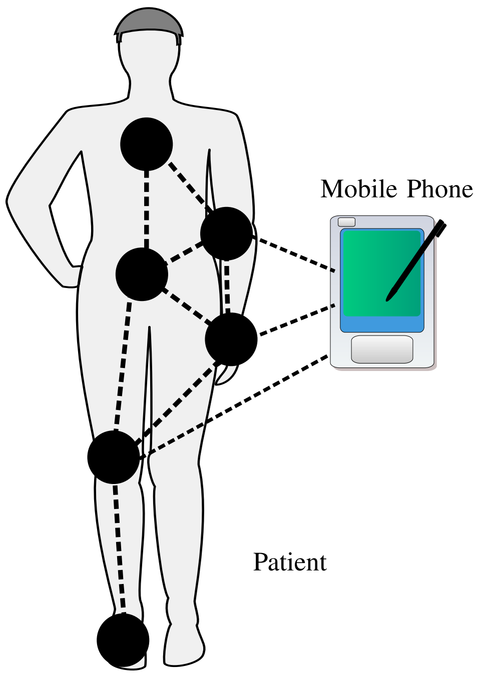

Fig. 1. Body Sensor Environment

Figure 1 gives a diagrammatic representation of a patient with body sensors and a mobile phone. The mobile phone is a specialized device that the patient will carry with them and it will be able to connect with the sensors and the internet. In the case where this system is part of a home health system, the mobile phone communicates with other devices in the home. Depending on the health risks and privacy concerns of the patient all of the information may not be transmitted to a hospital.

If a sensor detects an anomaly it can broadcast a command to all the sensors to increase their sampling rate. In the case where sensors do not have the intelligence to detects anomalies, the data is sent to the mobile phone. The mobile phone can then detect if there is anything wrong, if required then the mobile phone can broadcast a command to increase their sampling rate. Another important requirement is that sensors (especially sensors that are worn) on the body may be added or removed from the sensor network. The group key should be updated to handle additions and deletions of nodes to the sensor network.

II. Problems and Limitations

A. Group Key Protocols

One of the best known available group key protocol, which many other group key protocols are based on, is the Microtimed Efficient Stream Loss–Tolerant Authentication (μTESLA) protocol designed by Perrig et al. [2]. In μTESLA, when a message is broadcast, it generates a message authentication code for that message with a symmetric key. The key is a number on a *one–way hash chain* (OHC). Each sensor is preconfigured with the initial number of the OHC, called the *commitment*, so that when the key is released, each sensor node can verify whether the key is valid. Only one sensor (in most cases it will be the base station) will have the entire OHC. To securely broadcast a message, the base station sends the message and its message authentication code generated by the key. At a later time, the base station releases the key to all the sensors. The sensors will first verify that the key is on the OHC, if it is then it can confirm that the original message is correct.

There are several problems with this approach. The mechanism is inefficient if there are many broadcast messages lost, sensor nodes will then spend a long time verifying the key against the one–way hash function. Another problem is that when an important event occurs, and a command needs to be broadcast out to all the sensors to increase their sampling rates, the μTESLA delayed approach does not perform well. The command that was broadcast cannot be verified until at a latter time. And the final problem is that all the sensors will have to be preconfigured with the initial number of the OHC, which makes it difficult to add or replace sensors to the system.

Recent research in BSNs has shown that environmental information found in the body can be used to secure communication between sensors nodes [3].Health sensors can use Inter–Pulse–Interval (IPI) or Heart Rate Variance (HRV) as good sources for cryptographically random numbers and the physiological values can be used as a one–time pad. Protocols that used these physiological values to encrypt a new key between a sensor pair have been developed [4]. For instance, Venkatasubramanian et al. [4] used a single message to send a new key to the neighbouring sensor node, as shown in *Protocol 1*.

Protocol 1 Venkatasubramanian BSN protocol

$A \rightarrow B : N_A, [N_A]_K, K \oplus SEV$

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

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. However, the method used to generate a strong cryptographic key takes 30 seconds, since 67 quantized IPI values are required. Another problem was that physiological signals measured from different areas of the body have similar trends but not the same values.

Venkatasubramanian et al. [3] developed PSKA, as a mechanism such that the duration of the physiological signal capture is kept minimal. They used a *vault* [5] for the key agreement mechanism, where the phsiological data is used to lock and unlock the vault. However, the scheme requires approximately 50KB of memory and 10KB of transmitted data. Another limitation is that if a body sensor is lost or stolen, the contents of the vault are no longer secure. Some body sensors will not have enough resources, and may not be physically secured to be able to use the PSKA scheme.

Another mechanism to create a secure communication between body sensors is to place a small electrical charge around the body and use that as the communication medium [6]. However, the sensors themselves will become more complex as more components are added. Also, the network will become insecure if two body sensor networks were to come closer and intersect.

An approach for a secure group key generation for off the shelf sensors with minimal memory and bandwidth resources is required.

III. Creating a Group Protocol using SEVs

This paper uses the generic name *Secure Environmental Value* (SEV) referring to sensed data that can be obtained by sensors from their environment. This data is usually hard to obtain through other means. Examples of 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.
- 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. A major advantage of using SEVs is that they can be used for authentication in body sensors, since sensor on one's body will be measuring physiological data that is likely to be different from that is measured on another body at any given time.

Some of the data recorded from body sensors include the heart rate, blood pressure, temperature, and blood oxygen [7]. Table I shows the typical range of the data rate for each of the physiological values. The node information is 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). We have not included information such as the location of the receiving node in our node information, since it is a part of the packet header as described in TinyOS [8]. The heart rate values requires less than 2 bits per second, however, the node information increases the data rate to 10 bits per second. For the other physiological values including the node information, there is only a need for 2 to 3 bits per second. We assume the data rate of 10 bits per second, for measuring the time taken to execute our protocol.

TABLE I
BODY SENSOR DATA RATE RANGE

Signal Type	Depth	Node Info.	Msg Rate
Heart rate	8 bits	56 bits	10 msgs/min
Blood Pressure	16 bits	64 bits	2 msgs/min
Temperature	16 bits	64 bits	1 msgs/min
Blood oxygen	16 bits	64 bits	1 msgs/min

The major benefit of using environmental data is that body sensors can use the environmental data to authenticate that another sensor is also on the same body, and not a sensor on another individual.

Even though PINs and passwords may not be used feasibly in 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. Password protocols have the special property of allowing secrets with small entropy to be used for key establishment [9]. 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 SEV is compromised, after the execution of the protocol, then any session key that was created will not be compromised.

Key sizes in sensor networks are small, normally 64 bits, so that the encryption or integrity tests do not consume too much energy [2]. Small key sizes lead to the need to update keys on a regular basis. Another aspect of a heterogeneous sensor network is that different sensors measure different environmental data. There are also sensors that can measure more than one environmental phenomena.

A. A Group Protocol

If several entities sharing no previous secret need to create a shared group key, we can utilize a group key establishment protocol, such as the Burmester and Desmedt protocol as defined in *Protocol 2* [10]. The protocol was originally created without a formal proof, but subsequent work has shown the protocol to be provably secure [11]. The protocol provides both forward and backwards secrecy. Where an adversary who knows a set of group keys cannot derive any new group keys, or an adversary who knows a set of group keys cannot find any earlier group keys. The Burmester and Desmedt protocol uses a cyclic function to define the shared key. The protocol requires that the sensors are logically arranged in a ring so that $S_1 = S_{m+1}$, where m is the total number of sensors.

Protocol 2 Burmester–Desmedt generalised Diffie–Hellman with broadcasts

S_{i-1} $\qquad$ S_i $\qquad$ S_{i+1}

Round 1:

$$r_i \in_R \mathbb{Z}_p$$

$$\xleftarrow{t_i} \qquad t_i = g^{r_i} \qquad \xrightarrow{t_i}$$

$$X_i = (t_{i+1}/t_{i-1})^{r_i}$$

$$Z_{i-1,i} = t_{i-1}^{r_i}$$

Round 2:

S_i broadcasts X_i to all parties

S_i calculates Z

During Round 1 each adjacent pair of sensors, S_i and S_{i+1}, performs a basic Diffie–Hellman key exchange. The sensor S_i calculates the ratio, $X_i = (t_{i+1}/t_{i-1})^{r_i}$, from its two secrets from the adjacent sensors. In Round 2 each sensor broadcasts its X_i value. Once all the sensors have broadcast their values, every sensor can calculate the shared secret $Z = (Z_{i-1,i})^m X_i^{m-1} X_{i+1}^{m-2} \ldots X_{i-2}$.

The protocol has no authentication mechanism and by itself cannot be used in a sensor environment. When using this protocol with SEVs, we found that the difficulties of this protocol included :

- There is no authentication or key confirmation mechanism.
- Using weak keys to secure the protocol is non–trivial.
- There are $m+2$ exponents that are need to be calculated per sensor.

Since the protocol has no authentication mechanism, an adversary can easily join the protocol and discover the group key. We will show how SEVs can be used to authenticate all the participants in a group protocol, as long as every participant has access to the physiological values of the body (as an example) in real time. Key confirmation is also important to insure that all the participants in the group protocol obtained the newly created group key. We will show how SEVs can also be used as nonces.

Also, it is non–trivial using small keys (such as SEVs) to encrypt the communication. A security weakness occurs if the same weak key is used multiple times in the protocol. The same environmental data may be used over multiple protocol runs to conserve energy, or may be the environmental data stays constant over a period of time. When environmental data is used to encrypt t_i, an adversary can use brute force to decrypt $[[t_i]]_{SEV}$ over all the possible values of SEV and check if the candidate results are valid. If they are not then that particular SEV can be removed from the list of valid environmental values. If the same SEV is used multiple times, then the SEV can become compromised, and the adversary can join in the protocol and obtain the group key.

Encryption of data using a weak key introduces a brute force style attack in this protocol. An adversary guessing the SEV values can attempt to decrypt $[[t_i]]_{SEV}$ and examine whether the resulting plaintext is a valid Diffie–Hellman ephemeral value. For instance, if the decryption with a candidate SEV results in a string whose value is greater than p then that candidate is invalid.

Boyd and Mathuria [9] suggested an alternative method to calculate a shared secret when using a cyclic function. The method is to define $X_i = Z_{i,i+1} - Z_{i-1,i}$, and the way to calculate the shared secret is defined as Equation 1.

$$\begin{aligned} Z &= mZ_{i-1,i} + (m-1)X_i + (m-2)X_{i+1} + \ldots \\ &\quad +X_{i-2} \\ &= mZ_{i-1,i} + (m-1)(Z_{i,i+1} - Z_{i-1,i}) + \\ &\quad (m-2)(Z_{i+1,i+2} - Z_{i,i+1}) + \ldots \\ &\quad +Z_{i-2,i-1} - Z_{i-3,i-2} \\ &= Z_{1,2} + Z_{2,3} + \ldots + Z_{m,1} \\ &= g^{r_1 r_2} + g^{r_2, r_3} + \ldots + g^{r_m, r_1} \end{aligned} \tag{1}$$

However, there is a redundancy with Equation 1. The following is true $X_1 + X_2 + \ldots + X_n = 0$. This redundancy enables an adversary to make the protocol vulnerable to an attack by guessing the SEVs off-line and verifying whether the addition of decrypted values in the second round communication is zero. If so, the adversarys guess for SEVs is correct. Every participant needs to complete round 2 for this attack to be successful. Using SEVs to only encrypt the second phase does not yield a secure protocol and one can mount an off-line attack.

B. Proposed Group Protocol using SEVs

We now describe our proposed group key establishment protocol. It is based on the provably secure Burmester and Desmedt protocol. We have added authentication and key confirmation to this protocol using SEVs. We assume that the protocol has n sensors, where each sensor is able to sense the same environmental values as the other sensors in the system. The protocol is described in *Protocol 3*.

In the first round, each sensor S_i chooses a random time T_i where $T_i > currentTime$. The time will be used to notify all the participants at what time the sensor will be measuring its SEV. The time is set in the future so that an adversary cannot pick a time in the past, where they have calculated an old SEV. Each sensor will then broadcast out a message S_i, T_i informing all the other sensors at what stage they will calculate the SEV.

In the second round, each sensor will sense its environment and store the SEV V_i values for each sensor that is involved in the protocol. If two sensors are measuring the same SEV, the sensor with the larger id value will be given the next available time slot in the list. Insuring that $V_i \neq V_j \forall i, j$.

In the third round, we perform the Diffie–Hellman proportion of the protocol. However, before sending out t_i, we

Protocol 3 Singh et al. proposed group key protocol

S_{i-1} $\qquad$ S_i $\qquad$ S_{i+1}

Round 1:

S_i chooses a time T_i

S_i broadcasts S_i, T_i

Round 2:

S_i senses $V_1, \ldots, V_n$

Round 3:

$r_i \in_R \mathbb{Z}_p$

$t_i = g^{r_i}$

$\xleftarrow{t_i^*}$ $\quad t_i^* = t_i \oplus V_i \quad$ $\xrightarrow{t_i^*}$

$Z_{i-1,i} = t_{i-1}^{r_i}$

$Z_{i+1,i} = t_{i+1}^{r_i}$

$X_i = Z_{i+1,i} - Z_{i-1,i}$

Round 4:

S_i broadcasts $X_i \oplus V_i$ except S_n

S_i calculates X_n and then Z

Round 5:

S_i broadcasts $ENC_Z(S_i, V_i)$

encrypt it using V_i. Since each SEV is different, a partition attack is not possible.

In the fourth round each sensor sends $X_i \oplus V_i$, except for S_n. After all the messages have been broadcast, all the sensors will be able to calculate X_n from the equation $X_n = -(X_1 + X_2 + \ldots + X_{n-1})$. The reason to have the value X_n calculated by each sensor is to remove the redundancy found in Equation 1.

The fifth round verifies that each sensor has successfully calculated the new group key. It is important to add key confirmation at the end of this protocol, since an adversary may attempt a man–in–the–middle attack. This is done by each sensor encypting the values S_i, V_i with the group key Z, and then broadcasting the encrypted values to each of the other sensors in the group.

C. RSA and ECC Comparison

Previous research in sensors suggested that RSA is not suitable, and elliptic curves should be used [12], [13]. We extensively examine this claim and investigate if RSA can be efficiently implemented. To convert the RSA implementation of a password protocol into an elliptic curve implementation, the V_1 will need to be mapped to an elliptic curve point. The mapping requires a much larger key than the 160-bit exponent required for the above protocol. A general procedure for this can be found in *IEEE P1363.2: Standard Specifications for Password–Based Public–Key Cryptographic Techniques* [14]. A simplified version of the procedure is shown below.

1) Set $i = 1$.
2) Compute $w = h(A, B, V_1, i)$.
3) Set $\alpha = w^3 + aw + b$.
4) If $\alpha = 0$ then the point is $(w, 0)$.
5) Find the minimum square root of α, and call it β. Can use the method found in *IEEE P1363* [15] (Appendix A.2.5).

6) If no square root exists, set $i = i + 1$, and go to Step 2.
7) The elliptic curve point is (w, β).

The above algorithm is non–deterministic for different V_1 values. If a square root is not found then the algorithm will loop back to the second step to compute a new value for β. In an environment where a sensor scavenges energy to perform an operation, it is not suitable to have a non–deterministic algorithm.

Mapping a variable into a point on an elliptic curve allows the conversion of many RSA password protocols to an elliptic curve password protocol.

1) Mapping SEV into the field. When using RSA, the SEV can be naturally mapped into the field. This could either be a direct modulus, or a modulus of the hash of the SEV. However, mapping the SEV onto a point requires more work as shown by the following equation $P = f(A, B, V_1)r$. The function f is the non–deterministic method to map a number onto a point in an elliptic curve, as described above. The point is then multiplied by r, to place it into the correct group.
2) Both the RSA and the ECC implementation require a random value in the field $\mathbb{Z}_q$ to obtain an r_A as shown by $r_A \in_R \mathbb{Z}_q$.
3) The ephemeral value for Diffie–Hellman is calculated, for ECC, that is $t_A = r_A g$. The RSA algorithm involves an exponent.
4) Finally, the message is created. In the RSA case, P is multiplied to t_A, whereas in the ECC case the following equation is used $m = t_A + P$. The message m is created, to decrypt the message the receiver can subtract P, since $t_A = m - P$. An adversary will not be able to discover the value for t_A unless they know V_1. Also, the adversary will not find any invalid decrypted values, which removes the partition attack problem.

However, when converting this into an elliptic curve–based construct, we face the same problems as shown for the PAK family of products. The SEV will need to be mapped onto an elliptic curve point, which is non–deterministic.

We have developed a technique to make the protocol deterministic, where the elliptic curve point $(x, y) = (h_1(V_1, R_1), h_2(V_1, R_1))$. After this calculation, a valid elliptic curve is found where this point is valid $y^2 = x^3 + ax + b$, where the value for a is predefined. However, this does require extra encrypted information to be sent from A to B, $[[R_1]]_{V_1}, [[b]]_{V_1}$. In a resource constrained environment, extra information sent by a sensor will require additional energy to be consumed by the sensor.

IV. Analysis and Implementation

We implemented and compared the proposed group key agreement protocol on a Crossbow mica2 MPR2600 mote.The protocols were developed using TinyOS 2.x. The RSA code was based from the Deluge System, and we used a 160 bit exponent as required by most RSA based password protocols. Previous studies assumed that a larger key size is needed when using RSA [16]. However, password protocols have different characteristics where the assumption is not valid. We had one sensor attached to a workstation while the other sensors were placed stand–alone. After the running of the protocols, 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 configuration we used is shown in Figure 2. One of the sensors is attached to the computer with a USB cable. The computer registers the connection as a serial port. The communication between the computer and the sensor is achieved through the serial port.

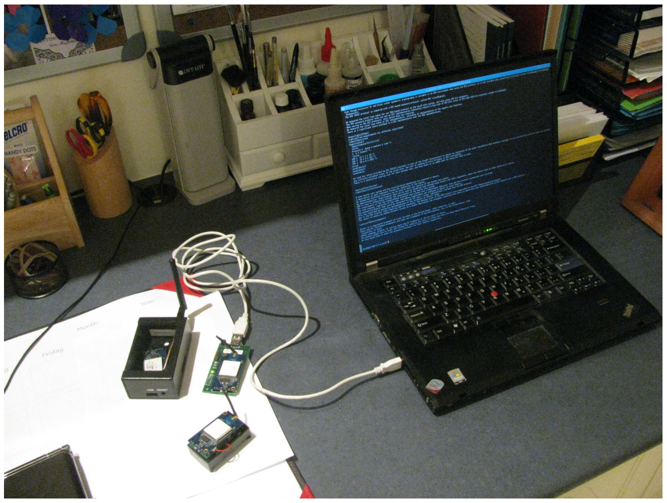

Fig. 2. Reading from the Sensor using the Serial Port

Table II shows the ratio of the application speed for each of the algorithms compared to exclusive-or algorithm. When we ran the algorithm on the mica2 mote, over the 2000 iterations it took approximately one millisecond. In the ATEMU simulator it took approximately 7000 instructions.

TABLE II
Comparison of Application Speed: As Ratio to exclusive-or

Algorithm	Mica2	ATEMU
RC5	453	456
SKIPJACK	739	741
HMAC–MD5	18400	18500
RSA	41600	41900
SQRT	87800	88400
ECC	4820000	4920000

We found little difference between the simulation results and the amount of time an operation takes when put on the mica2 mote. The most notable difference in results was for the ECC algorithm where there was a two percent difference. Both the time and number of instructions suggest that for the same size key the RSA algorithm is significantly better than the ECC algorithm.

A. Memory Size

The size of the application both in terms of number of lines of code and the size in bytes is important when choosing an algorithm. Table III list the number of lines of code and the size in bytes of the application that we used to run our original time and instruction measurements.

TABLE III
MEMORY SIZE FOR DIFFERENT ALGORITHMS

Algorithm	Code Lines	Size (bytes)	Stack (bytes)	RAM (bytes)
XOR	80	5600	158	432
RC5	506	6776	172	466
SKIPJACK	697	8138	190	496
RSA	1456	7062	213	624
SQRT	3366	7662	230	748
ECC	5038	14020	760	2066

The *Code Lines* indicates lines of code and thus the complexity of the code for a developer to implement the application. The *Size (bytes)* indicates the size in bytes of the application. The *Stack (bytes)* indicates the maximium size of the stack for the application. The *RAM (bytes)* is the maximium amount of RAM the application will need. The figures are obtained from the stack analysis tool found in tinyos.

The XOR application is the quickest by several orders of magnitude compared to the other cryptographic primitives. But 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.

We found that the performance costs of the cryptography algorithms was insignificant compared to the costs of communication. Where the communication can be measured in the milliseconds the computation costs had to be measured in the microseconds range. On average each message sent took 10 ms to be received and parsed by the opposite end. When we ran the protocol over four nodes, we found that the total time took was 97 ms. We have yet to try this protocol with a larger number of sensors.

We also investigated the memory used by the protocol. We developed an application without RSA. The application only sent and received messages. We then added the parsing and calculations of the group key into the code. We found that the application without RSA had a maximum RAM size of 411 bytes. By adding the RSA algorithm into the application the RAM size grew to 624 bytes.

V. Conclusions and Future Work

In this paper, we proposed a secure group key establishment mechanism for a body area network. It allows multiple devices to agree on a shared group key, in an authenticated manner, without the need for any form of initialization or a priori knowledge. We showed that designing a secure group key agreement protocol is not a trivial task, and discussed different types of attacks. The proposed group protocol uses SEVs to achieve authentication. The protocols were implemented in TinyOS and run on micaz motes. The time elapsed and memory usage showed that the protocol meets the general BSN requirements. We show that a group key establishment protocol can be efficiently run in a sensor network with the RSA algorithm. Future work includes further performance testing of our protocol with a larger number of sensors.

References

[1] K. Singh and V. Muthukkumarasamy, “Verification of key establishment protocols for a home health care system,” in *Proceedings of the Fourth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, Sydney, Australia, December 2008.

[2] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, “SPINS: Security protocols for sensor networks,” in *Seventh Annual International Conference on Mobile Computing and Networks (MobiCOM 2001)*, Rome, Italy, July 2001.

[3] K. K. Venkatasubramanian, A. Banerjee, and S. K. S. Gupta, “Pska: Usable and secure key agreement scheme for body area networks,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 1, pp. 60–68, 2010.

[4] 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.

[5] A. Juels and M. Sudan, “A fuzzy vault scheme,” in *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*, 2002, p. 408.

[6] T. Falck, H. Baldus, J. Espina, and K. Klabunde, “Plug ’n play simplicity for wireless medical body sensors,” *Mob. Netw. Appl.*, vol. 12, pp. 143–153, March 2007.

[7] T. Balomenos, “User requirements analysis and spcification of health status analysis and hazard avoidance artefacts,” DC FET Project ORESTELA, Delieverable D02, Tech. Rep., 2001.

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

[9] C. Boyd and A. Mathuria, *Protocols for Authentication and Key Establishment*, U. Maurer and R. Rivest, Eds. Springer Berlin / Heidelberg, 2003.

[10] M. Burmester and Y. Desmedt, “A secure and scalable group key exchange system,” *Inf. Process. Lett.*, vol. 94, no. 3, pp. 137–143, 2005.

[11] J. Katz and M. Yung, “Scalable protocols for authenticated group key exchange,” *Journal of Cryptology*, vol. 20, pp. 85–113, 2007.

[12] R. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, and P. Kruus, “Tinypk: securing sensor networks with public key technology,” in *SASN ’04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*. New York, NY, USA: ACM Press, 2004, pp. 59–64.

[13] D. J. Malan, M. Welsh, and M. D. Smith, “A public–key infrastructure for key distribution in tinyos based on elliptic curve cryptography,” in *Proc. 1st IEEE Communications Society Conference on Sensor and Ah Hoc Communications and Networks (SECON ’04)*. Santa Clara, CA, USA: IEEE Computer Society Press, October 2004, pp. 71–80.

[14] IEEE, “Standard specifications for password–based public–key cryptographic techniques,” IEEE, Tech. Rep. IEEE 1363-2000, 2006, http://grouper.ieee.org/groups/1363/passwdPK/draft.html.

[15] ——, “Standard specifications for public key cryptography,” IEEE, Tech. Rep. IEEE 1363-2000, 2000, http://grouper.ieee.org/groups/1363/P1363.

[16] H. Wang and Q. Li, “Efficient implementation of public key cryptosystems on mote sensors (short paper),” in *Information and Communications Security*, ser. Lecture Notes in Computer Science, P. Ning, S. Qing, and N. Li, Eds. Springer Berlin / Heidelberg, 2006, vol. 4307, pp. 519–528.