A Light-Weight Secure Broadcast for delay-tolerant applications in Single-Hop Wireless Sensor Networks

Hassan Nasiraee, Jamshid B. Mohasefi
Dept. of Computer Engineering
Urmia University
Urmia, Iran
hnasir20@gmail.com , j.bagherzadeh@urmia.ac.ir

***Abstract*—Symmetric encryption of data using time-varying keys at base-station is proposed as an attractive and interesting method for broadcasting in wireless sensor networks since last time. In this paper, we have proposed a light-weight encryption scheme using time-varying at cost of delayed-verification. Main idea of the proposed model is that instead of using one key per each packet, we use one key per each given number of packets. However, a significant problem is that interference or disconnections may cause a receiver to miss broadcast packets and the dynamic keys contained therein, rendering it unable to participate in subsequent broadcasts. Hence, we improve our proposal by a patch-like loss-recovery scheme. At the end, we will discuss the security and performance aspects of our method.**

Keywords-Security; secure broadcast; time-varying keys; efficiency ; wireless sensor network

I. INTRODUCTION

Wireless sensor networks use network broadcast in various applications such as software update and network management. A sensor can also send data to the base-station (BS), through unicast, and the BS in turn broadcasts it to all sensors. Security of message broadcasting is an important and basic worry, especially in scenarios that data are critical, such as military applications. It is easy to eavesdrop and inject false data in wireless sensor networks. Traditional solutions designed for point-to-point networks, cannot be directly applied in sensor networks [12].

Sensor networks have severe resource constraints and they might be combined of hundreds or thousands of nodes. Using a simple symmetric cryptographic approach has some problems, e.g., if a node is compromised and its key is revealed, the attacker can masquerade and sniff all network traffic. In contrast, using an asymmetric cryptographic mechanism is not suitable in sensor networks for the sake of high computational overhead. There is a need to a light-weight time-varying scheme that satisfies most important security metrics. In a time-varying key cryptographic method broadcast messages are encrypted with different keys which make a chain of keys.

In the typical time-varying key models, a hash function is used to generate keys chain. To build a keys chain, hash value of a seed is calculated more and more to generate an array of keys. This key is used to encrypt data in reverse order periodically. The last generated hash value is used as the first or root key, which is inserted into all sensor nodes in early stages of network deployment. In the first stage of communication, BS encrypts a new key with root-key and then broadcasts it to all nodes. Sensors would decrypt the received packet with root-key and extract new key from it, then they authenticate the new key by comparing its hash value with the root key. The next keys will be transmitted in the same manner.

The problem with this method is that using a key for each packet, means tolerating communication overhead of one key and computation overhead of one hash function per any one packet. In our method proposed in this paper, we use only one key per any given number, as £, of packets.

In fact, at this paper we are seeking a light-weight secure broadcast scheme along with ensuring source authenticity, confidentiality, DoS resistance and simplicity. We name the proposal as LWSB, which stands for Light-Weight Secure Broadcast.

A. Assumptions

Albeit our scheme is general for most communication networks, but for simplicity in discussion, we assume a single-hop wireless sensor networks. Such a network has a BS that can broadcast data directly to sensor nodes. Like most of the works in the field, we assume that the BS has enough computation and energy resources. If we pay more cost and equip nodes with Tamper-Resistant hardware such as Tamper Platform Module (TPM) [16], security of key materials in sensor node will be improved significantly. We assume the wireless channel is open and unsecure. In this channel, an adversary can eavesdrop; inject false data to the network and replay packets.

B. Paper Organization

The rest of this paper is organized as follows: Previous works will be described in section II. In section III we have proposed our solution. Fault-tolerance is discussed in section IV. In section V, we analyzed its performance. In addition, we discuss experimental results in this section. We conclude in the section VI and highlight once again the salient features of our scheme.

II. PREVIOUS WORKS

Albeit many works accomplished in asymmetric cryptography field such as [8], but for the sake of computational and energy consumption limitations, most

researches in sensor networks focus on symmetric cryptography [1,6]. In the following, we review some well-known secure broadcast scheme in symmetric cryptography.

One of the early protocols which are used for secure communication in sensor networks is TinySec [4]. In this method, the symmetric secret key that is used for cryptography is shared among all sensors in the network and has long lifetime. Although TinySec has good advantages and it is simple but it is unsecure. An attacker can potentially find the key over time. In addition, a compromised node can easily transmit forged messages. This protocol is vulnerable against replay attack also.

MiniSec [7] operates in two modes, broadcast and unicast. This scheme is similar to TinySec except some modifications such as an extra counter to prevent replay attacks.

µTesla [11] and X-TESLA [18] are used to broadcast authenticated data without considering confidentiality. Network-wide loose time synchronization, buffer space for buffering packets, and vulnerability against replay attacks in each interval are main defects of this scheme.

Multi-level µTesla [5] is another scheme that is proposed to improve µTesla with simplified key distribution phase and uses multi-level keys.

LEAP [17] uses a broadcast authentication protocol such as µTesla, which consequently has loose-time synchronization and delayed verification problems.

In [15] the authors have proposed SOSJ (stands as acronym of author's names) for confidential and secure broadcasting in wireless sensor networks with Time-varying keys. In this scheme, the BS generates a hash chain with successive hashing (RC5, SHA-1 or MD5 [3]) which starts from a seed value such as K (which we name it K^M for convenience). This chain is like $K^M, K^{M-1}, \dots, K^i, \dots, K^1, K^0$ where the key K^{i-1} is the hash value of the key K^i. It is assumed that the keyK^0, which is the last key generated in the hash chain, is inserted in any sensor node securely in the network deployment phase or by Diffie-Hellamn protocol. In this protocol, a key is encrypted with a private key and would be decrypted with the same key. The $(i+1)^{th}$ key will be inserted in the i^{th} packet and the packet will be encrypted with the i^{th} key and will be sent to all sensors. In addition to have many advantages regarding security aspects, using one key for each packet, means tolerating communication overhead of one key and computation overhead of one hash function per any one packet.

III. PROPOSED METHOD

To create a broadcast session key between BS and any sensor, we insert an initial key as commitment in any sensor (this key is shared in all sensor nodes) securely. This key would be inserted, either in node manufacturing time or after network deployment, and can be updated over time. In our proposal, we use one key per any given number, as £, of packets. The j^{th} session key to decrypt the £ consecutive packets is a key which is received from BS in i^{th} ($i = £ \times j$) packet. The key, K_j , is made in the BS in a hash chain similar to µTesla method. By successively hashing a seed value like K_3^M , we generate the sequence of keys as $K_M, K_{M-1}, \dots, K_j, \dots, K_1, K_0$. We insert K_0 in all sensors as initial key and keep the rest keys in the BS ($1 \le j \le M$).

Any sensor verifies authentication of new key,K_j , by checking if its hash value yields the previous key, K_{j-1}. A key in the key chain is transmitted from the BS to any sensor, per £ packet. The value £ is a fixed number greater than 0 which is calculated based on various conditions and environment parameters in network deployment phase. If we determine £ equal to 1, then the BS like in SOSJ must send a key with every packet which is not required in many delay-tolerant application.

We have two message types. First type includes data and MAC, as Message Authentication Code, for authentication. Second type includes data and an encryption key, as K_j. The MAC value in type1 messages are made by K_j.

To inhibit forge message from sender and guarantee the source authentication, we first make (£-1) type 1 messages by K_j, as a key to encrypt them and making included MAC value. Then we broadcast them to the receivers. An attacker cannot forge the type1 messages, because are not aware about K_j. Sender after broadcasting (£-1) type 1 messages; broadcast the corresponding type 2 messages which includeK_j.

In fact, we are using time varying keys instead of constant keys. To this purposed, we use reverse of a hash chain with long step. Which means, per a given number of encryption packets (£ number), encryption keys will be updated, only one time. The sequence of encryption keys and corresponding broadcast packets are shown at below. The sequence of encryption keys:

$$K_0 \leftarrow K_1 \leftarrow K_2 \leftarrow \cdots \leftarrow K_{\frac{M}{£}-1} \leftarrow K_{\frac{M}{£}}.$$

The sequence of broadcast packet:

$$P_0 \leftarrow P_1 \leftarrow \cdots \leftarrow P_£ \leftarrow P_{£+1} \leftarrow \cdots \leftarrow P_{M-£+1} \leftarrow \cdots \leftarrow P_{M-1} \leftarrow P_M.$$

So key $K_{\frac{M}{£}}$ is used to broadcast £ consecutive broadcast packet, $P_{M-£+1} \leftarrow \cdots \leftarrow P_{M-1} \leftarrow P_M$, in our proposal.

To inhibit replay attack among any £ consecutive packets which use same key to encryption/decryption, we use same counter in both sides, as sender and receiver.

To reduce the effects of key compromising within a sensor's memory (probably by physical intrusion) we suggest using a group rekeying method [9,10]. Any sensor should have a unique shared key with the BS (in addition to the broadcast key) that will be used for private conversation between the sensor and the BS. To repel or decrease effects of physical intrusion attack, when a node is compromised and this event is detected by the BS, in any way, the group rekeying protocol would be established (usually this event is discovered by neighbors of the compromised node and informed to the BS). This process removes compromised node from group and network conversations by transmitting new group keys (current encryption key) to all nodes except the compromised one.

This scheme inherits some attributes of asymmetric cryptographic methods, due to using asymmetry mechanism for authentication in hash chains.

It is resistant against replay attacks, by using the counter. But TinySec is weak against replay attacks.

Our scheme has enough resistant against overflow attack but the methods based on μTesla are vulnerable against it. Our method acts in a different way and is not based on time interval, delayed verification, and does not use buffer for this purposes.

A. Applications, Extensions and Generality

Main point of our work, as a confidential authenticated broadcast, is a general method to provide confidential authenticated broadcast stream over channels. The LWSB may be used in other applications which use secure communication and secure broadcast as a separate building block. For example, LWSB may be used in secure routing [11,12] and multicast multimedia stream over channels. One of the best applications of our proposal is delay-tolerant networks, which use delay-tolerant applications. A delay and disruption-tolerant network, DTN, is an occasionally connected network that may suffer from frequent partitions and that may be comprised of more than one divergent set of protocols or protocol families. The DTN network is utilized in various operational environments, including those subject to disruption and disconnection and those with high-delay; the case of deep space is one specialized example of these, and is being pursued as a specialization of this architecture (see [24,25, RFC 4838] for more details). Other networks to which the DTN architecture applies includes sensor-based networks using scheduled intermittent connectivity, terrestrial wireless networks that cannot ordinarily maintain end-to-end connectivity, satellite networks with moderate delays and periodic connectivity, and underwater acoustic networks with moderate delays and frequent interruptions due to environmental factors [26].

B. Generalization for unicast communication

We can generalize the described method for unicast communication between BS and a sensor node. Its advantage versus broadcast model is, if one or some sensors are compromised, other sensors are still safe and secure. BS uses this communication model to transmit high confidential data to sensors.

IV. Fault-Tolerant LWSB

We use a patch-like method to tolerate loss in our scheme that is inspired from [9]. The method is called PKB (Periodically Key Broadcast).

Disrupting and losing packets containing a key will disturb our scheme and will make it unreliable (similar to the schemes that carry keys inside packets such as [15]). To build our method fault-tolerant and resistant against packet loss, we use PKB as follows. In PKB we use a master key (K_{MS}) that is inserted into node in the network deployment time. In PKB the last encryption key (K_j) that is sent by the BS is encrypted by K_{MS} and is broadcasted into the network, periodically. We assume one out of r packets containing the key, contains the recovery information.

We want to show that extending LWSB by PKB with ratio r would be more reliable. It means, the probability of a node to miss the key, in a long time, is near to zero. Before explaining the issues, we introduce some parameters as follows:

n: Number of recent keys which have not received yet,

r: Ratio of PKB packets to packets with the key,

P_{fault}: Probability of missing a packet containing the encryption key

$P_{WTK,n}$: Probability that a node has not received last n encryption keys

The relation between the parameters is defined as:

$$P_{WTK,n} = (P_{fault})^{r \times n}. \quad (1)$$

With respect to practical values of the parameters, the value of $P_{WTK,n}$ would be too small. For example if $n = 5$, $P_{fault} = 1/2$, and $r = 1$ then

$$P_{WTK,5} = (1/2)^{1\times 5} = 0.03125. \quad (2)$$

Obviously we can assume that for $n > 5$, the value of $P_{WTK,n}$ is near to zero and consequently PKB is reliable.

A. Overhead of PKB

Because of the unlimited power assumption for BS, overhead on BS is not sensible and is tolerable. However, sensors would be affected more than BS because of their resource constraints. We calculate this overhead in the following. We show the length of encryption key and length of data portion as K_{len} and D_{len}, respectively. With respect to practical values of the parameters as below:

$$D_{len} = 15\ byte,\ K_{len} = 16\ byte,\ r=1,\ £ = 10. \quad (3)$$

We have:

$$Overhead_{node} = r \times K_{len} / (£ \times D_{len} + K_{len}) = 8 / 166 = 0.0482. \quad (4)$$

Having the above overhead on decryption cost and packet receiving cost, we guarantee reliability of our scheme with an acceptable approximation.

For environments with rare packet loss, we decrease the overhead on sensor, BS, and communication channel by decreasing the value of r. For example if we decrease r with coefficient 5, then r will be 1/5 and we have:

$$Overhead_{node} = r \times K_{len} / (£ \times D_{len} + K_{len}) = (1/5) \times 8 / 166 = 0.00964. \quad (5)$$

V. Performance Evaluation

To encrypt/decrypt packets with session keys and generating the MAC value of type 1 packet, we use RC5 for the sake of its high speed execution and admissible security. In addition

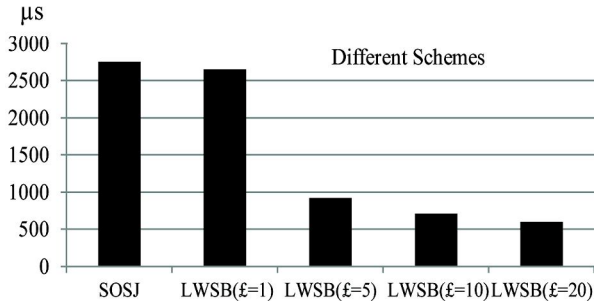

Figure 1. Comparing average computation time for any packet in sensor for different schemes

its optimized version has a good performance on 8 bit micro controllers [3,6,14]. To generate hash chain we use SHA-1 as our hash function. We assume 16-byte key length and 4-byte for MAC length.

Here we would like to find a mathematical relation for average computation time of receiving, processing and preparing the next session key in the scheme as LWSB and SOSJ. The computation times for these two methods are called T_{LWSB}, T_{SOSJ}, respectively. The parameters that are used in relations, described as follows:

T_h: Hash function execution time per byte,
T_D: Cryptographic function execution time that is used to decrypt received packet, per byte,
£: Network parameter,
D_{Len}: Number of bytes of data portion of the received packet,
K_{Len}: Number of bytes of key portion in the received packet,
MAC_{Len}: Number of bytes of MAC portion
T_{MAC}: Execution time to compute MAC value by RC5 per byte,
T_{LWSB}: Average computation time for any packet of the method LWSB,
T_{SOSJ}: Average computation time for any packet of the method SOSJ,

Note that T_{LWSB} depends on different actions including the decryption of the received packet, calculation of the MAC value and calculation of the hash key (if included within the received packet). This value is calculated by the following formula:

$$T_{LWSB} = \left(D_{len} + \frac{K_{len}}{£}\right) T_D + MAC_{len} T_{MAC} + \left(\frac{K_{len}}{£}\right) T_h. \quad (6)$$

The computation time of the method SOSJ, T_{SOSJ}, depends on the decryption time of the received packet, and the calculation time of the hash value which exists within the received packet:

$$T_{\mathrm{SOSJ}} = (D_{len} + K_{len}) T_D + (K_{len}) T_h. \quad (7)$$

By selecting proper values for the above parameters, we have compared LWSB ($£ = 1, 5, 10, 20$) against SOSJ for processing time required in a sensor node.

Like SOSJ and most works in the area we have selected SHA-1 as our hash function. We consider all conditions (for example node computational power and so on) as same as those in [3]. They have assumed 26 µs for encrypting one byte with RC5 and 122 µs for one byte hash value computation with SHA-1. With considering the above descriptions, we have:

$$T_D = T_{MAC} = 26\,\mu s\,. \quad (8)$$

$$T_h = 122\,\mu s\,. \quad (9)$$

With this description we have:

$$T_{LWSB} = 806 + 1392/£\ \mu s. \quad (10)$$

$$T_{SOSJ} = 1782\ \mu s\,. \quad (11)$$

Fig.1 compares T_{ISBU} and $T_{\mathrm{SOSJ}}$with respect to different values of £. Out of the 20 bytes generated by SHA-1, only the lower 16 bytes were used by RC5 for encryption/decryption. In our calculations we have ignored non-sensible times like increasing or decreasing a counter.

Therefore our method has clear efficiency in processing time. As Fig .1 shows, the processing time in our proposal for the case $£ > 5$ desired efficiency. Though methods like TinySec need less computation but they don't have security of LWSB or SOSJ. For example, TinySec does not use reverse hash chain to repel packet forging. Consequently, they do not have hash function computation which is the expensive processing time in a sensor node.

LWSB does not have any problem in energy consumption aspects. Energy consumption in a node is due to sending and receiving data rather than computation. When sending and receiving, LWSB concatenates a 16-byte key to the data portion once per £ packets but in SOSJ for any one packet a key must be concatenated to the data. Thus for $£ > 1$, the amount of energy consumption in LWSB is less than SOSJ. In memory consumption, we only need the required memory for RC5 and SHA-1 primitives.

A. Experimental Results

To evaluate network broadcast traffic, overhead on communication channel and memory overhead, we have prototyped LWSB and SOSJ on Tmote sky [22,2] to perform a real world scenario comparison. For ease of implementation, we have used available source codes from TESLA [27].

The experiments are done on a PC (which runs Cygwin) as a BS which programmed by free cryptographic APIs from Bouncy Castle JCE provider [21]. Like [1,15], we did not prototype these two schemes as extension of Deluge in TinyOS, as our proposal is loss-tolerant and do not need reliability mechanism of Deluge [23]. In fact, we extended TinyOS standard packet for our prototypes. These schemes have been installed in Tmote [19].

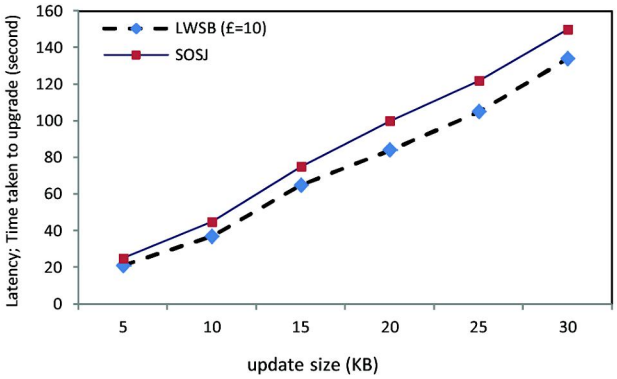

Figure 2. Latency comparison for upgrade in different schemes

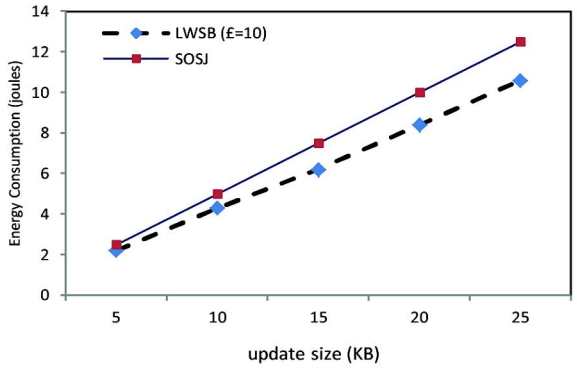

Figure 3. Energy consumption comparison in different schemes to update a sample program (25 KB)

These prototypes are implemented in a network comprised of 10 Tmote sky nodes (with MSP430 F1611 microcontroller) [22] and a PC as a BS.

In the experiments, we have used five programs to profile the time taken to update them and to measure memory usage. These five programs are Blink (6 KB), Pong (11 KB), TinyECC (23 KB), secure-Deluge (39 KB) and a sample program (50 KB). We have done each experiment several times and only considered consistent outputs disregarding error-bars.

To conduct a comparison based on energy consumption, we have used power TOSSIM [20] on Mica-2, which is a well-known tool in WSNs. Since there is an exhaustive energy models for Mica-2 such as [20], we have evaluated the energy consumption analysis for Mica-2 in our work.

We performed an experiment, to evaluate the time of upgrading a sensor node versus the size of information (broadcasted). In the two schemes, the upgrade time and the gap between any two curves is increased by increasing the updated information size. In fact, the latency in these schemes depends on the update size and is independent of number of nodes, as shown in Fig. 2.

As it can be seen, latency in LWSB is a less than SOSJ, as discussed before.

In our analysis, we evaluated energy consumption a node based on updated information size. We have used five programs to profile the energy consumption based on updated information size in the schemes (LWSB and SOSJ). These five programs, with different sizes, are Blink, Pong, TinyECC, secure-Deluge and a sample program. We have done each experiment several times and only considered consistent outputs disregarding error-bars.

In this experiment, we compared the energy consumption of the schemes. In the schemes, any sensor receives packets independent of other sensors (despite multi-hop programming). In fact, there is a direct linear relation between energy consumption and size of updated information (Fig. 3). As it can be seen in the figure, energy consumption in LWSB is less than SOSJ, as discussed before.

VI Conclution

In this paper, we proposed a lightweight secure broadcast scheme. We have shown its efficiency on communication and computation overhead. Network broadcast schemes for wireless networks that use time-varying keys en/decryptions are susceptible to key loss due to wireless medium. Hence, we used a patch-like key loss recovery scheme. It is lightweight, secure, scales easily and is highly configurable. A receiver does not have to reveal its location by transmitting explicit key requests. It only listens to the BS to make its session keys. To evaluate security of the proposed scheme, we compared it with a number of famous schemes in the area, against some prevalent attacks in sensor networks.

References

[1] S. T. Ali, V. Sivaraman, A. Dhamdhere, D. Ostry, “Secure key loss recovery for network broadcast in single-hop wireless sensor networks,” Ad Hoc Networks, vol. 8, no. 6, pp. 668-679, Aug 2010.

[2] Crossbow Technologies, Mica2 and MicaZ motes, <*http://www.xbow.com*>.

[3] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F.Mueller, M. Sichitiu. “Analyzing and Modeling Encryption Overhead for Sensor Network Nodes”. In: WSNA‘03, San Diego, USA, September 2003.

[4] C. Karlof, N. Sastry, D. Wagner, “TinySec: a link layer security architecture for wireless sensor networks,” In: SenSys’04, ACM,Baltimore, Maryland, November 2004.

[5] D. Liu, P. Ning, “Multilevel µTesla: broadcast authentication for distributed sensor networks,” ACM Transactions on Embedded Computing Systems 3 (4) (2004) 800–836.

[6] J.Lopez , J.Zhou ,*WIRELESS SENSOR NETWORK SECURITY* , Ebook, IOS Press, 2008

[7] M. Luk, G. Mezzour, A. Perrig, V. Gligor, “MiniSec: A secure sensor network communication architecture,” in: International Conference on Information Processing in Sensor Networks, ACM/IEEE, Cambridge, Massachusetts, April 2007.

[8] Sk.Md. M. Rahman,K. El-Khatib , “Private key agreement and secure communication for heterogeneous sensor networks “, J. Parallel Distrib. Comput. 70, pp. 858-870, 2010 .

[9] D. Naor, M. Naor, and J. Lotspiech. “Revocation and tracing schemes for stateless receivers.” In Advances in Cryptology - CRYPTO 2001. Lecture Notes in Computer Science, vol. 2139. 41–62. 2001

[10] A. Perrig, D. Song, and D. Tygar. “Elk, a new protocol for efficient large-group key distribution”. In Proc. of IEEE Symposium on Security and Privacy. 2001.

[11] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, “SPINS: security protocols for wireless sensor networks,” in: Mobile Computing and Networking, ACM, Rome, Italy, 2001.

[12] A. Perrig and J. D. Tygar, “Secure Broadcast Communication in Wired and Wireless Networks.” Kluwer Academic Publishers, 2002.

[13] D. R. Raymond and S. F. Midkiff, ″Denial-of- Sevice in Wireless Sensor Networks: Attacks and Defenses,″ IEEE Pervasive Computing, vol. 7, no. 1,pp. 74-81, 2008.

[14] R. Rivest, “The RC5 encryption algorithm,” in: Proceedings of Workshop on Fast Software Encryption, pp. 86–96, 1994.

[15] J. Shaheen, D. Ostry, V. Sivaraman, S. Jha, “Confidential and secure broadcast in wireless sensor networks”, in: IEEE International Symposium for Personal, Indoor and Mobile Radio Communications, PIMRC, Athens, September 2007.

[16] Trusted Computing Group, Trusted Platform Module Specifications, <*http://www.trustedcomputinggroup.org/specs/TPM/*>.

[17] S. Zhu, S. Setia, S. Jadojia, “LEAP: efficient security mechanisms for large-scale distributed sensor networks,” in: CCS’03, ACM, Washington, DC, October 2003.

[18] T. Kwon and J. Hong, ″Secure and Efficient Broadcast Authentication in Wireless Sensor Networks, ″ Computers, IEEE Transactions on, vol. 59, no. 8, pp. 1120-1133, Aug 2010.

[19] “Telosb-Telosb Mote Platform,” http://www.willow.co.uk/ TelosB_Datasheet.pdf, Sept. 2010.

[20] V. Shnayder, M. Hempstead, B.-R. Chen, G.W. Allen, and M.Welsh, “Simulating the Power Consumption of Large-Scale Sensor Network Applications,” Proc. Int’l Source Conf. Embedded Networked Sensor Systems (SenSys ’04), pp. 188-200, 2004.

[21] Bouncy Castle Crypto Apis, http://www.bouncycastle.org, 2010.

[22] Tmote Sky, “Ultra Low Power IEEE 802.15.4 Compliant WirelessSensor Module Humidity, Light, and Temperature Sensors with USB,” http://www.moteiv.com/products/docs/tmote-sky-data sheet.pdf, Aug. 2007.

[23] J. Hui, D. Culler, “The dynamic behavior of a data dissemination protocol for network programming at scale,” in: SenSys, ACM,Baltimore, Maryland, 2004.

[24] S. Burleigh, et al., "Delay-Tolerant Networking - An Approach to Interplanetary Internet", IEEE Communications Magazine, July 2003.

[25] InterPlaNetary Internet Project, Internet Society IPN Special Interest Group, http://www.ipnsig.org.

[26] F. Warthman, "Delay-Tolerant Networks (DTNs): A Tutorial v1.1", Wartham Associates, 2003. A vailable from http://www.dtnrg.org.

[27] A. Perrig, R. Canetti, D. Song, and D. Tygar, “Efficient Authentication and Signing of Multicast Streams over Lossy Channels,”In:Proc. IEEE Security and Privacy Symp., Mar. 2000.