Investigating the Feasibility of LEAP+ in ZigBee Specification

Mohammad Rezaeirad, Muhammad Aamir Iqbal, Dmitri Perkins, Magdy Bayoumi
The Center for Advanced Computer Studies
University of Louisiana at Lafayette
Lafayette, LA 70504-4330
Email: {rxm1725, mxi1678, perkins, mab@cacs.louisiana.edu}

***Abstract*—The ZigBee specification is an emerging wireless technology designed to address the specific needs of low-cost, low-power wireless sensor networks and is built upon the physical and medium access control layers defined in IEEE 802.15.4 standard for wireless personal area networks (WPANs). A key component for the wide-spread success and applicability of ZigBee-based networking solutions will be its ability to provide enhanced security mechanisms that can scale to hundreds of nodes. Currently, however, an area of concern is the ZigBee key management scheme, which uses a centralized approach that introduces well-known issues of limited scalability and a single point of vulnerability. Moreover, ZigBee key management uses a public key infrastructure. Due to these limitations, we suggest replacing ZigBee key management with a better candidate scheme that is decentralized, symmetric, and scalable while addressing security requirements. In this work, we investigate the feasibility of implementing Localized Encryption and Authentication Protocol (LEAP+), a distributed symmetric based key management. LEAP+ is designed to support multiple types of keys based on the message type that is being exchanged. In this paper, we first conduct a qualitative security analysis of LEAP+ and the current ZigBee key management scheme. Using the QualNet 5.0.2 simulator, we implement LEAP+ on the ZigBee platform for the very first time. Experimental results show that a distributed key management scheme such as LEAP+ provides improved security and offers good scalability.**

***Index Terms*—ZigBee, IEEE 802.15.4, Key management, LEAP+**

I. Introduction

Wireless sensor networks (WSNs) comprise small battery powered and low cost devices each with sensing, data processing and communication capabilities. ZigBee is an emerging standard that aims to address applications in a wide range of markets, including commercial building automation, residential appliance networks, health-care, fitness, telecommunication, and even military, police, safety, and rescue applications [1]. In ZigBee networks, a large number of sensor nodes are deployed to monitor a wide area, where the working situations are commonly tough. Since these nodes are typically positioned in distant locations and they might have mission critical tasks, they should be armed with security appliances to provide information assurance against any unwanted information leakage [2]. Unfortunately, the constraints of WSNs make them more vulnerable to attacks including denial of service (DOS), traffic analysis, and node replication. Even jamming mitigation techniques are not generally feasible in WSNs to use against DOS due to their design complexity and high energy consumption [3], [4].

The ZigBee specification includes a number of security provisions and options [5] while improving the basic security framework defined in IEEE 802.15.4 [6]. ZigBee security service facilitates carrying out secure communications, establishing of cryptographic keys and controlling devices. Indeed, key management is a core mechanism for any other security services in ZigBee protocol stack. The objectives of this key management are to generate and securely distribute required cryptographic keys between the communicating nodes that need to transfer data. Unfortunately, ZigBee asymmetric based key establishment is not efficient (e.g., Diffie-Hellman key establishment protocol [7]) due to energy consumption and hardware requirements [8], [4]. Sufficient security cannot be provided by ZigBee when large sensor networks are employed [9]. ZigBee key management scheme is not too flexible for node addition and revocation while working in any desired environments. Moreover, ZigBee key management is not meant for distributed application due to its centralized design. This results in a need to find an efficient and reliable candidate scheme to replace ZigBee key management to overcome such limitations and security issues. Hence, in this work, we select LEAP+ [10] to be implemented as an alternative to ZigBee key management scheme. Using decentralized key management such as LEAP+, however, has its own costs to be paid to satisfy anticipated security requirements. Our contributions include: first, a detailed security analysis and comparison of the LEAP+ and ZigBee key management schemes; second, the implementation and integration of LEAP+ in the ZigBee protocol stack as an alternative to the standard ZigBee key management.

The remainder of the paper is organized as follows. Section II provides an overview of ZigBee specification and its security. In Section III, LEAP+ is briefly described as a replacement key management scheme followed by Section IV, where we analyze and compare both the key management schemes. We describe our implementation of LEAP+, designed scenarios in QualNet and performance evaluation due to this replacement of key establishment protocol on ZigBee in section V. Finally, in Section VI, we draw conclusive remarks and suggest future work.

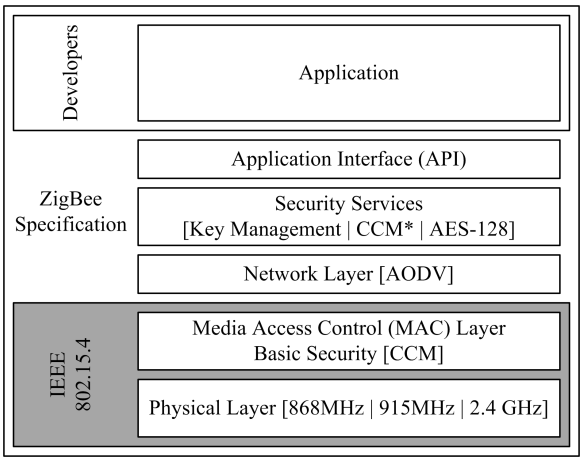

Fig. 1. The ZigBee protocol stack

II. ZigBee Specification and Security

ZigBee specification is considered a reliable, low-power, wirelessly networked monitoring and control product by the ZigBee Alliance [5]. ZigBee comprises IEEE 802.15.4 for the physical and MAC layers along with its support for network and application layers and places itself on top of the IEEE 802.15.4 layers as it is shown in Fig. 1. The IEEE 802.15.4 standard applies to WPAN that operate at low data rate wireless connectivity and are confined to operate in the range of 10m [6].

ZigBee supports various levels of security that can be configured depending on the needs of the application. It includes methods for key establishment, key transport, frame protection, and device management [5], [11]. ZigBee provides three types of security modes: residential, standard, and high security. Residential security is first supported in the ZigBee 2006 standard [5]. This level of security requires a network key to be shared among devices and is designed for lower security residential applications. Standard security adds a number of optional security enhancements over residential security, including an application support sub-layer (APS layer) link key. High security (commercial) adds entity authentication, and a number of other features not widely supported. This mode is intended to be implemented for high security commercial applications. ZigBee high security utilizes three types of keys: master key, link key, and network key. The master key is used for secure communication between nodes and the base station. The link key is shared by two devices for secure unicast communication where as the network key is used for broadcast communications and is shared among all devices in the network, both of these types of keys can be updated periodically. The base Station (Trust Center device) authenticates devices that are going to join the network. This Trust Center takes care of the link key distribution in the network and also selects a proper network key. All the devices therefore, must be pre-configured with the proper link key to enhance the network security.

Fundamentally, all the keys are delivered via either pre-installation, key-Transport, or key-Establishment methods as defined by ZigBee [5]. In the pre-installation method keys are loaded before placement in the network. In the Key-Transport technique, the Trust Center transmits the key in a secure fashion to the device whenever possible. Any node may obtain its network key via key-transport or pre-installation. In the Key-Establishment approach, the link key establishment is processed by a symmetric-key key establishment (SKKE) protocol [12] and it is based on master key. The master key itself is acquired via key-transport or pre-installation [12]. The network key is securely transported between the Trust Center and the device by using a link key based on the 128-bit Advanced Encryption Standard (AES) encryption algorithm [13]. First, the Trust Center encrypts the network key by using a link key and sends the encrypted data; then the device decrypts the received data by using the link key. The master key is a secret key between two nodes and provides a starting point for establishing a link key. This task can be done via other mechanisms such as Certificate-based Key Establishment (CBKE) and Alpha-secure Key Establishment (ASKE) [11].

TABLE I
Key management security requirements vs. ZigBee key management specification

Requirements	Zigbee
Availability	Entire of the network can be compromised and unavailable due to its centralized structure if there is an attack on the base station.
Authentication	Only the master key is used for authentication when the high security feature is enabled that decreases the level of security considerably.
Confidentiality	Support AES-128 as encryption algorithm.
Integrity	Uses AES-CCM* and ECDSA Digital Signatures.
Non-repudiation	Good
Connectivity	High global and low local connectivity.
Degradation of security services	Three different types of security profiles have been implemented.
Efficiency	Uses the asymmetric cryptosystem hence it has a bad impact on performance of WSNs.
Flexibility	Good
Revocation	Not supported well.
Scalability	Limited to 254 nodes only.
Survivability	Not very robust against node capture attacks.

Besides all the security specifications of ZigBee, WSNs have similar type of security requirements to those of ad-hoc networks [14], [15]. There are general and specific security requirements for any key management of WSNs as discussed in [16], [17]. Table I illustrates how ZigBee supports these security requirements and features. Clearly, ZigBee key management does not support some of these requirements well. One issue is the centralized nature of ZigBee key management. If the Trust Center is compromised then the whole network becomes compromised since it has a single point of exposure. Another issue is the scalability of the network. ZigBee cannot supply sufficient security when large sensor networks appear. Such shortcomings give a motivation to use any other key

management protocol such as LEAP+. In section IV, we discuss the properties of ZigBee and LEAP+ key management schemes in more detail.

III. LEAP+

LEAP+ is a distributed key management protocol for WSNs that is designed to support the establishment of four types of keys depending on the type of messages that are being exchanged [10]. It also provides an efficient protocol for local broadcast authentication based on the use of one-way key chains. This authentication protocol supports source authentication without preventing passive participation where neither the globally nor pair-wise shared keys are used for authentication. In the next subsections, the four types of key establishment methods are described:

A. Establishing Individual Node Keys

Every node has its own unique preloaded key IK_a (where a is a unique node ID) that is shared with the base station to be used for secure communication and is generated as $IK_a = f_{K_m}(a)$, where f is a pseudo-random function and K_m is a master key known only to the base station.

B. Establishing Pair-wise Shared Keys

This key is shared by every node with its immediate neighbors for secure communications and source authentication. Once a new node is added, its initial network key, K_{IN} is used to derive the master key and the pair-wise keys shared with its neighbors. For adding M nodes to a sensor network in M intervals of $T_1, T_2, T_3, ..., T_M$, the base station arbitrarily produces M keys as $K_{IN}^1, K_{IN}^2, K_{IN}^3, ..., K_{IN}^M$ respectively serving as initial keys. There are four steps required to add a node a to the network to establish a pairwise key with each of its neighbors described as follow:

1) Pre-distribution: A node a, loaded with the initial key K_{IN}^i, derives its master key as $K_a^i = f_{K_{IN}^i}(a)$ for the current time interval T_i, and loaded with master keys $K_a^j = f_{K_{IN}^j}(a)$ for all future intervals of $i < j \leq M$, but not any initial keys K_{IN}^j matching to those time intervals.

2) Neighbor Discovery: The node a broadcasts a $HELLO$ message containing its ID and the interval i and a timer starts after T_{min} time (we assume there exists a lower bound on the time interval T_{min} that is necessary for an adversary to compromise a sensor node, a newly deployed sensor node should discover its immediate neighbors within this time interval). Every neighbor b responds with an ACK message including its ID. This ACK from each neighbor b is authenticated via current master key K_i^b of node b. As node a knows K_{IN}^i, it derives K_i^b to verify the ACK message from every neighbor.

3) Pair-Wise Establishment: Node a generates its pair-wise key K_{ab} with b as $K_{ab} = f_{K_b^i}(a)$. In similar way, node b might also generate K_{ba}.

4) Pair-Wise Establishment: The timer that is started after a $HELLO$ message was broadcast, stops after time T_{min}, node a deletes K_{IN}^i and all the master keys K_i^b of its neighbors. Node a does not delete its own master key K_i^a or any other preloaded master keys K_j^a where $i < j \leq M$.

C. Establishing Cluster-Keys

Every node a shares a unique cluster key with all its immediate neighbors $b_1, b_2, ..., b_m$ for secure message broadcasting, followed by the pair-wise key establishment phase. Node a begins producing a random key K_a^c, then encrypts this key with the pair-wise key shared with each immediate neighbor, and then sends the encrypted key to each neighbor b_i, $i < j \leq M$. In a similar way, each node b_i stores the key K_a^c in a table after decrypting, and sends its own cluster key to node a.

D. Establishing the Global-Key

This secret key is shared by all nodes in the network and the base station and is used for secure global broadcasting such as revocation announcements. There must be a broadcast authentication mechanism, since any adversary may claim to be the base station by broadcasting un-authenticated messages. μTESLA [18] protocol has been used for broadcast authentication because of its efficiency and tolerance to packet loss. Let a is the node to be revoked, k'_g new global key and k_i^T is to be disclosed μTESLA key. The base station broadcasts the following message M:

$$M : BS \rightarrow * : a, f_{k'_g}(0), MAC(k_i^T, a \mid f_{k'_g}(0)) \quad (1)$$

The base station broadcasts the MAC key k_i^T after one μTESLA interval. Node b authenticates received message M after receiving MAC key that arrives one μTESLA interval later. After a positive authentication, node b saves the verification key $f_{k'_g}(0)$ temporarily. And if that node b happens to be neighbor of node a, node b erases its pair-wise key shared with a and updates its cluster key.

IV. Analysis

In this section, we draw a comparison of different features of original ZigBee's key management scheme and LEAP+ that is given in Table II. ZigBee's key management scheme is a centralized scheme that relies on a Trust Center having both public and private keys whereas LEAP+ is a distributed key management scheme having a symmetric key as its main cryptosystem. Obviously, asymmetric cryptosystem does not scale well in a large network that consists of hundreds of devices. However, this scalability issue is addressed in LEAP+ by eliminating the distribution center for key management [19].

ZigBee provides three types of keys that are preloaded before node deployment [5]. Both link and network keys can be updated either manually or online. ZigBee does not have a proper mechanism to transmit the master key to each node in a secure fashion resulting in utilizing public-key system technology that incorporates performance overheads. LEAP+ provides four types of keys where individual and global keys are preloaded before deployment [10]. Both the pair-wise and cluster keys are generated and established after node deployment and discovery of its own immediate neighbors. ZigBee does not support a practical node revocation mechanism. Therefore, if a node is captured, the keys might become

TABLE II
COMPARISON BETWEEN ZIGBEE AND LEAP+ KEY MANAGEMENT SCHEMES

Specification	ZigBee Key Management	LEAP +
Network Architecture	Centralized	Distributed
Cryptosystem	Public and Private Key	Private Key
Scalability	Limited	Very Good
Initial Key Transmission	Not Safe	Secure
Type of keys	Master, Link and Network Key	Individual, Cluster, Pair-wise and Global-key
Authentication	Global Broadcast Authentication	Global and Local Broadcast Authentication
Mode of operation	CCM*	CBC-MAC
Key update	Periodically	Event Driven
Re-keying policy	Not Well Defined	Enforced

available to the adversary. The ZigBee re-keying policy is not well-defined even for the Smart Energy Profile [5], [11] and it does not have proper re-keying mechanisms, increasing security and efficiency concerns. In LEAP+, all four type of keys can be revoked and updated via LEAP+'s revocation and re-keying mechanism where the re-keying and revocation policies are comprehensively defined and enforced.

Unlike ZigBee that offers global broadcast authentication, LEAP+ supports both local and global broadcast authentications without preventing passive participation initially inherited from μTESLA. Employing local broadcast authentication has its own performance benefits, especially where the event- or time-driven messages can be locally authenticated such as routing control messages or aggregated sensor readings. Having local broadcast authentication mechanism eliminates potential associated costs that can be introduced by global broadcast authentication [10]. Routing control messages or aggregated sensor readings are examples of event- or time-driven local broadcasts that do not impose delay and energy overheads as global broadcasts usually have. ZigBee uses the CCM* [20] mode of operation, which is a general combined encryption and authentication block cipher mode with a block size of 128-bit such as AES-128. To extend it to other block sizes requires further definitions again increasing the performance overhead whereas CBC-MAC [21] is used in LEAP+ where block size is fixed to offer authentication. Thus, a key management scheme that scales a large flexible network with decentralized controller is found in LEAP+, making it an obvious potential choice to replace ZigBee's key management scheme.

V. IMPLEMENTATION AND PERFORMANCE EVALUATION

This section describes how the cryptographic functions presented in LEAP+ are implemented on top of the ZigBee stack as part of application layer. The LEAP+ Key management

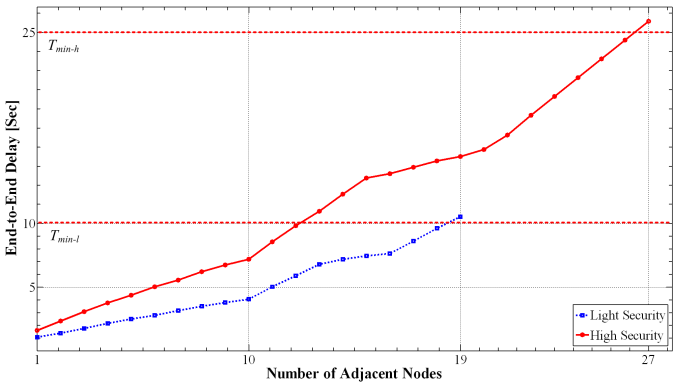

Fig. 2. Time delay for a node to establish pair-wise keys with all its neighbors for the high and light security configurations

scheme routines are coded in C++ and introduced as part of simulation functions of QualNet.

LEAP+ requires some specific cryptographic functions such as key generation, message authentication and encryption. To meet these requirements, RC5 is used as the random key generation and encryption function. Respectively, Cipher-based Message Authentication Code (CMAC) is utilized to check integrity and confidentiality of data and to authenticate the communicating entities [22], [23], [21]. RC5 is appropriate for WSNs applications due to its low memory requirement. Moreover, the RC5 block cipher has embedded parameter variability to get flexibility at all levels of security and efficiency. RC5 is also employed as part of CMAC implementation. We have configured two levels of security (high and light security) by modifying data-dependent rotations (via increasing RC5 rounds from 16 to 64) to be used in our designed scenarios.

We performed experiments for three different scenarios. These experiments are replicated ten times for each scenario. The nodes are Fully Functional Device (FFD), immobile and randomly placed within the areas of 20×20, 25×25, 30×30 and 50×50 square meters.

In the first scenario, a node establishes a unique pair-wise key with each node within its communication range (immediate neighbors). In each step, we deploy a new node within this range. We continue adding new nodes up to the point that the time delay reaches the T_{min} threshold. We adopt two different thresholds. When high security is required, T_{min-h} is the maximum tolerable time delay allowed by the protocol for the nodes to establish their pair-wise keys, and T_{min-l} is used when higher performance is required while still maintaining a minimum level of security. We also adopt the value of 10 seconds for T_{min-l} based on the performed experiment in [10], and we assume the value 25 sec for T_{min-h}. Note that, these values can be decided differently (based on application requirements) since the time to establish pair-wise keys is far less than the time for an adversary to obtain copies of all the memory and data on the captured sensor node.

Our main objectives of this experiment are to indicate the maximum number of nodes that can be placed within the

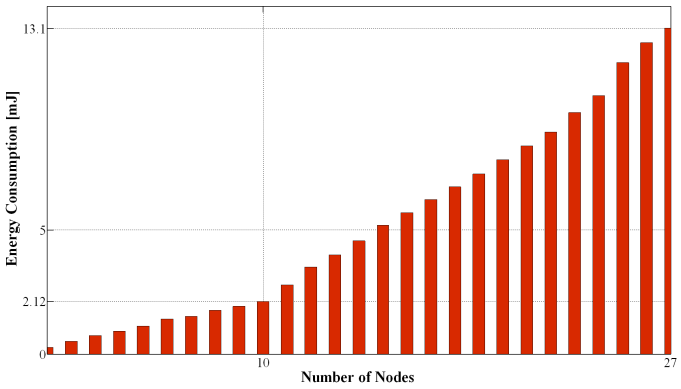

Fig. 3. Energy consumption for a node to establish pair-wise keys with all its neighbors

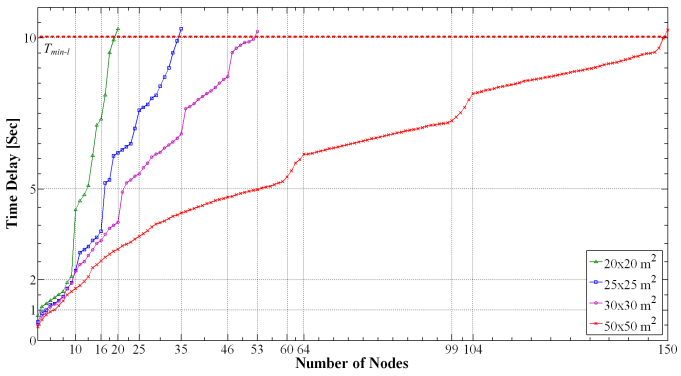

Fig. 4. Time delay for the entire network to establish pair-wise keys for the light security configuration

radio range of the aiming node, and the impacts of network density on both time delay and energy consumption. The total number of nodes n within the area of communication range r of the aiming node defines the node density $d = \left(\frac{n}{\pi r^2}\right)$. Fig. 2 depicts the time that is taken for added nodes (27 nodes in high security and 19 nodes for light security) to generate and deliver pair-wise keys to a target node within one cluster. Correspondingly, energy consumption due to these key transmissions is shown in Fig. 3. It is observed that when the number of nodes in a network is increased, more time is taken for a node to make a pair-wise key with the other neighboring nodes and also more energy is consumed for transmissions and communications accordingly. The simulation statistic file shows that for the first ten nodes, the total number of Bytes sent by the sending node is identical to received Bytes by receiving nodes; this confirms that there is no congestion and as a result the graph inclines linearly. Immediately after adding more nodes to this cluster, we observe that the time delay increases exponentially. This is due to retransmission of sending packets in some cases up to five times. This introduced packet loss ratio is mainly caused by network congestion. At the same point, energy consumption grows at an altered rate, as shown in Fig. 3. Moreover, monitoring the physical status of the targeting node shows that it spends over 40% of time in either Transmit and Receive mode, rather than Idle or Sleep mode.

To demonstrate the scalability of LEAP+, we expanded the first scenario. We ran the simulation for different numbers of nodes and areas of network. In contrast from the previous observation, we consider time delay for the entire network due to pair-wise key establishment. First, we begin establishing pair-wise keys for a network size of 20×20. Once we reach the time threshold, we stop adding nodes to the network. We then ran the same simulation for a larger network size of 25×25. As soon as we hit the thresholds, we proceed with network sizes of 30×30 and 50×50 square meters. Fig. 4, exhibits the time taken for the entire network to obtain the corresponding pair-wise Keys. When operating in an area of 20×20 square meters with the light security configuration, the frequency spectrum gets congested when the number of nodes reaches 19. Delay due to this congestion can put the key establishment protocol in a compromised state. The only way to add more nodes to the network is to provide more room by increasing the network area, thereby decreasing average network density. However, we have to keep the average density to a reasonable amount where the network is still connected (we enforce this in the simulation, by not having any isolated nodes). Also, the average density of the network must be less than or equal to the node density calculated from the first scenario, in order to maintain the pair-wise key delay always less than the time threshold. Let n be the total number of nodes in a cluster, A be the network area, and r be the communication range of a single node. The expected maximum admitted nodes N to the network can be calculated as:

$$N \leq n\left(\frac{A}{\pi r^2}\right) \tag{2}$$

For instance, given a network region of 30×30 square meters and radio range of 10 meters, the cluster size is 19 nodes where nodes are positioned uniformly. The total number of admitted nodes to the network should not be more than 54 nodes for the light security configuration. For the same setup, our simulation result confirms, that the network size of 53 nodes does not exceed the threshold of 10 sec (see Fig. 4). The high security configuration anticipates more delays while providing more security over the light security setup. Fig. 5 shows simulation results for this setup; these delays might not be interesting for some applications where the performance aspects of key management require more attention. Furthermore, having a higher time threshold in the high security setup allows more node admission to the network that increases the overall scalability and connectivity of network.

As mentioned in Section III, cluster-key is used for secure broadcast within a cluster. Cluster formation happens right after pair-wise key establishment and these cluster-keys are securely transported by pair-wise key encryption. In the second scenario, we validate the operation of cluster-key establishment procedure. In this setup, for all nodes added to the network in first scenario, a unique cluster-key is established. The extra cost of this secure mechanism (cluster-key establishment) is calculated in term of time delays and energy consumption.

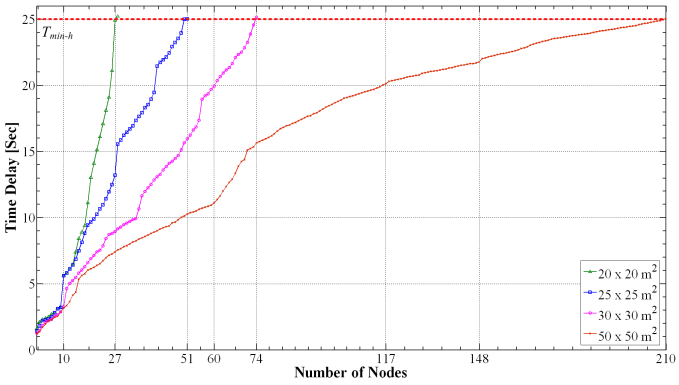

Fig. 5. Time delay for the entire network to establish pair-wise keys for the high security configuration

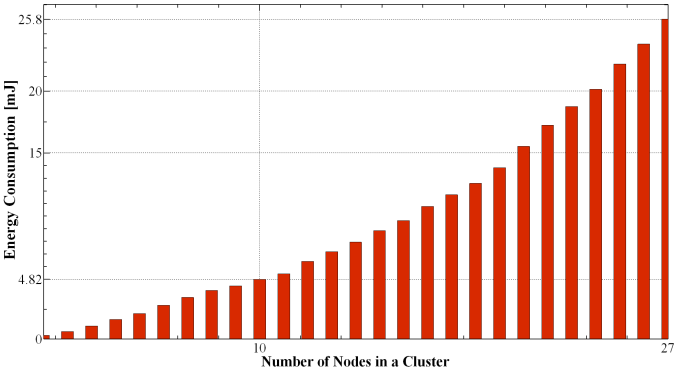

Fig. 7. Energy consumption for all the nodes within a cluster to generate and exchange their cluster-keys

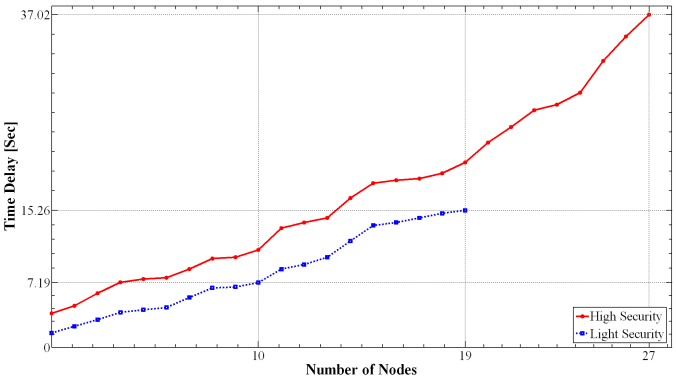

Fig. 6. Time delay for all the nodes within a cluster to generate and exchange their cluster-keys for the high and light security configurations

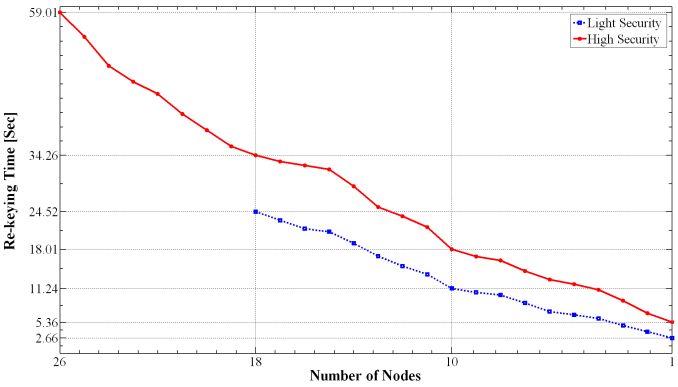

Fig. 8. Time delay for nodes within a cluster to update and re-establish keys (re-keying) with the other nodes for the high and light security configurations

Fortunately, LEAP+ did not fix the time threshold for the key delivery and left this option open for application developers to decide. It is evident from Fig. 6 that the total time delays to establish cluster-keys increases with regard to the number of nodes being increased in a network and it is the same behavior for the energy consumptions as described in Fig. 7.

Increasing the number of nodes impacts the end-to-end delivery and energy consumption for the same reasons described for the first scenario. The only difference is that there is additional time and energy taken by the encryption and decryption for the cluster-keys transmissions. The cluster-keys are transported securely (encrypted with pair-wise key), therefore, there will be several unicast secure sessions between each set of immediate neighbors for secure exchanging of the cluster-keys.

Finally, in the third scenario, compromised nodes are consecutively revoked from the aforementioned cluster. Re-keying procedures subsequently are performed to stabilize the connectivity of the network by recovering disconnected paths. When a node is revoked, all nodes that are neighbors of the revoked node need to encrypt their new cluster-keys using the pair-wise key shared with each neighbor. Therefore, the number of such secure key transportations are determined by the number of neighbors and the density of the sensor network. Fig. 8 shows the total time taken to revoke a node from the network. If a node is revoked from a cluster that comprises 27 nodes, there is a delay of 4.37 seconds at most for the high security profile. This time is taken to inform all of the 26 remaining nodes and let them authenticate the revocation message coming from the base station. For a network with reasonable density, it seems that transmission time delays do not cause many performance problems in LEAP+. For example, for a network of 26 nodes, the total re-keying time increased from 59.01 seconds as depicted in Fig. 9. All of this happens within a single cluster. Therefore, if there is a need to deploy additional nodes, LEAP+ provides the possibility to increase the number of clusters instead of overcrowding a single cluster as was argued in previous scenarios.

We evaluate the computation and communication costs of the LEAP+ key establishment schemes for each of the aforementioned scenarios. This paper does not provide a quantitative comparison among ZigBee key management and LEAP+ schemes based on mentioned costs. The reason is that both these protocols have different fundamental characteristics. First, in a distributed design such as LEAP+, there is no single building block for key establishment, whereas in a centralized architecture the main controller organizes key establishment. Therefore, the method of key distribution will be different and cannot be compared within the same scenarios. Second, both pair-wise keys and cluster-keys in LEAP+ are established after

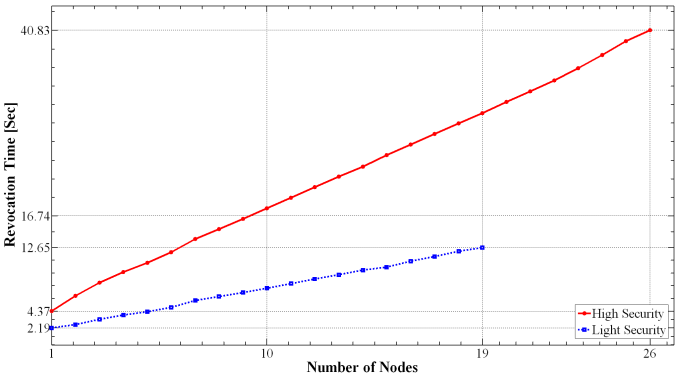

Fig. 9. Time delay for all the nodes within a cluster to revoke a node for the high and light security configurations

node deployment whereas link and network keys in ZigBee are preloaded. Also, LEAP+ keys can be regenerated and reestablished securely after authentic revocation by the base station as in our third scenario. The achieved results help network application developers to have a clear picture about both security features and additional communication overheads introduced by LEAP+. Inspecting Table II, it is possible to see the benefits of LEAP+, but these have associated costs which are examined in this section.

VI. Conclusion

In this work, we substitute the key management scheme of ZigBee by implementing LEAP+ to enhance the security capabilities. LEAP+ is a symmetric distributed key management protocol for sensor networks that is designed to support multi-type keys depending on the type of message that is being exchanged. In fact, LEAP+ forms the network into overlapping small clusters providing the possibility to have better security by reducing the risks of information leakage that are caused by broad information exchanged. LEAP+ is surprisingly well-suited to different types of network topologies, device types, and addressing modes offered by ZigBee stack. Our experimental results and performance evaluation parameters are not only valuable to assess the feasibility of LEAP+ scheme on the ZigBee protocol stack but they also provide the basis for having an effective mechanism to get reasonable scalability within WSNs. There is, however, a significant point to be considered. That is, LEAP+ is essentially meant for stationary nodes. Mobility of the nodes within the network is highly significant for mobile wireless sensor networks and needs to be considered for future work. This need inspires the idea to upgrade LEAP+ with mobile capability, keeping in mind that there are a lot of design challenges and potential issues that must be addressed and resolved in order to enable mobility in sensor networks to get enhanced security and reduced performance overheads.

References

[1] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer Networks*, vol. 51, pp. 921–960, 2007.

[2] J. Lee, V. Leung, K. Wong, J. Cao, and H. Chan, "Key management issues in wireless sensor networks: current proposals and future developments," *Wireless Communications, IEEE*, vol. 14, no. 5, pp. 76 –84, october 2007.

[3] N. Gura, A. Patel, A. W, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and rsa on 8-bit cpus," 2004, pp. 119–132.

[4] A. S. W, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," 2005.

[5] Z. Alliance, "Zigbee specification," ZigBee Alliance, Tech. Rep., June 2007.

[6] "IEEE standard for information technology- telecommunications and information exchange between systems- local and metropolitan area networks- specific requirements part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (wpans)," *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, 2006.

[7] W. Diffie and M. E. Hellman, "New directions in cryptography," 1976.

[8] A. Hegland, E. Winjum, S. Mjolsnes, C. Rong, O. Kure, and P. Spilling, "A survey of key management in ad hoc networks," *Communications Surveys Tutorials, IEEE*, vol. 8, no. 3, pp. 48 –66, qtr. 2006.

[9] B. Zhang and L. Chen, "An improved key management of zigbee protocol," in *Proceedings of the 2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, ser. IITSI '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 416–418.

[10] S. Zhu, S. Setia, and S. Jajodia, "Leap+: Efficient security mechanisms for large-scale distributed sensor networks," *ACM Trans. Sen. Netw.*, vol. 2, pp. 500–528, November 2006.

[11] G. Dini and M. Tiloca, "Considerations on security in zigbee networks," in *Proceedings of the 2010 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, ser. SUTC '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 58–65.

[12] H. Kim, J.-M. Chung, and C. H. Kim, "Secured communication protocol for internetworking zigbee cluster networks," *Comput. Commun.*, vol. 32, pp. 1531–1540, August 2009.

[13] *Advanced Encryption Standard (AES)*, Federal information processing standards (FIPS 197) Std., November 2001.

[14] L. Zhou and Z. Haas, "Securing ad hoc networks," *Network, IEEE*, vol. 13, no. 6, pp. 24 –30, 1999.

[15] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Security Protocols*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2000, vol. 1796, pp. 172–182.

[16] S. A. Camtepe and B. Yener, "Key distribution mechanisms for wireless sensor networks: a survey," Rensselaer Polytechnic Institute, Tech. Rep., 2005.

[17] Y. Xiao, V. K. Rayi, B. Sun, X. Du, F. Hu, and M. Galloway, "A survey of key management schemes in wireless sensor networks," *Comput. Commun.*, vol. 30, pp. 2314–2341, September 2007.

[18] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins: security protocols for sensor networks," *Wirel. Netw.*, vol. 8, pp. 521–534, September 2002.

[19] D. Christin, P. S. Mogre, and M. Hollick, "Survey on wireless sensor network technologies for industrial automation: The security and quality of service perspectives," *Future Internet*, vol. 2, no. 2, pp. 96–125, 2010.

[20] D. Whiting, R. Housley, and N. Ferguson, "Counter with cbc-mac (ccm)," United States, 2003.

[21] B. Preneel, "Cbc-mac and variants," in *Encyclopedia of Cryptography and Security*, H. Tilborg, Ed. Springer US, 2005, pp. 63–66.

[22] R. R. Mit and R. L. Rivest, "The rc5 encryption algorithm." Springer-Verlag, 1995, pp. 86–96.

[23] *Recommendation for Block Cipher Modes of Operation:The CMAC Mode for Authentication*, NIST, U.S. National Institute of Standards and Technology Std., May 2005.