

SECK: Survivable and Efficient Clustered Keying for Wireless Sensor Networks

Michael Chorzempa and Jung-Min Park
Bradley Department of Computer and Electrical Engineering
Virginia Tech
Blacksburg, VA 24061
{mchorzem, jungmin}@vt.edu

Mohamed Eltoweissy
Department of Computer Science
Virginia Tech
Falls Church, VA 22043
eltoweissy@vt.edu

Abstract

A wireless sensor network (WSN) typically consists of a large number of small sensor nodes and one or more high-end control and data aggregation nodes. Sensor nodes have limited computation and communication capabilities, and communicate via wireless links. Consequently, WSNs are highly vulnerable to attacks. This vulnerability is exacerbated when WSNs have to operate unattended in a hostile environment, such as battlefields. In this paper, we propose a novel self-organizing key management scheme for large-scale WSNs, called Survivable and Efficient Clustered Keying (SECK). Our scheme was designed specifically to address the key management issues within the low-tier of a hierarchical network architecture. Previous approaches for WSN key management adequately addressed operational issues, but to a large extent, ignored robustness and recoverability issues. Using simulation and analysis, we show that SECK is highly robust against key and node captures, and has noteworthy advantages over other key management schemes.

Keywords: Secure Wireless Sensor Networks, Self-Organizing Security Protocols, Key Management, Network Clustering, Exclusion Basis System.

1. INTRODUCTION

Key management is crucial to the secure operation of Wireless Sensor Networks (WSNs). A large number of keys need to be managed in order to encrypt and authenticate all sensitive data exchanged. We propose an efficient solution for group key management, called Survivable and Efficient Clustered Keying (SECK), that is appropriate for a network with a multi-tier hierarchical architecture. Our scheme is designed to manage keys within the bottom tier of such a network. We give more details on the assumed network architecture in Section 2. SECK is a self-organizing scheme that establishes pairwise keys, provides efficient methods for distributing and maintaining session keys, and provides mechanisms to recover from node captures in a hostile environment. Using analytical and simulation results, we show that SECK is flexible for

various models of communication, and is robust against the attacks that we identify in this paper. Moreover, our results show that SECK incurs low communication and computational overhead on the sensor nodes.

In section 2, we describe the overall WSN architecture that is assumed throughout the paper. We describe the fundamental design principles of SECK in Section 3, present the threat model in Section 4, and give full details of SECK in Section 5. In Section 6, we discuss SECK's communication overhead and its robustness against node captures. In Section 7, we describe related research. Finally, we summarize our findings in Section 8.

2. THE NETWORK ARCHITECTURE

In a flat network, all nodes are identical and there is no predetermined architecture. Although simple and efficient for small network sizes, the flat network architecture lacks scalability. A WSN with a multi-tier architecture strikes a compromise between cost and performance that is not possible with a flat architecture. In this section, we describe a two-tiered network architecture that is suitable for large scale WSNs.

In this two-tier architecture, a small number of high-end nodes, called Aggregation and Forwarding Nodes (AFNs), are deployed together with numerous low-end sensor nodes, called Micro Sensor Nodes (MSNs). In addition, the network includes a globally trusted base station, which is the ultimate destination for data streams from all the AFNs. The base station has powerful data processing capabilities, and is directly connected to an outside network. Each AFN is equipped with a high-end embedded processor, and is capable of communicating with other AFNs over long distances. A MSN is a battery-powered sensor node equipped with a low-end processor and mechanisms for short-range radio communications at low data rates.

The bottom tier of the network consists of multiple clusters, where each cluster is composed of numerous MSNs clustered around an administrative AFN. We assume in a typical network deployment, where every MSN

is within 2 hops of its local AFN. The top tier consists of AFNs and the base station.

Within each cluster, a set of keys need to be deployed and managed to secure transmissions between the MSNs and the AFN. SECK was designed for this very purpose.

3. FUNDAMENTAL DESIGN PRINCIPALS OF SECK

In this section, we explain the fundamental design principles employed in SECK by describing a basic stripped down instance of the scheme. In particular, we focus our discussions on the way SECK manages the administrative keys within a cluster.

To distribute and refresh session keys, SECK assigns a set of administrative keys to each node in a network. To manage administrative keys within a cluster, SECK employs the Exclusion Basis System (EBS) [5]. The EBS is essentially a mechanism for assigning the administrative keys of each node. The EBS is defined by $EBS(n, k, m)$ where n is the number of users supported in the system, k is the number of keys within each key subset, and m is the number of keys from the global key set not held within a subset. We denote the global administrative key set as U , where $|U| = k + m$. A given EBS supports $\binom{k+m}{k}$ unique subsets of keys. For full details of the EBS, see [4,5]. We denote the i -th node N_i , and its assigned administrative key subset is denoted S_i , with $|S_i| = k$. We also denote the keys not held by N_i as T_i , where $T_i = U - S_i$. A subset of the global key set is uniquely assigned to each node such that the remaining nodes each have at least one of the keys not assigned to that node, that is, for all $j \neq i$, $S_j \cap T_i \neq \emptyset$. This property is the main motivation for utilizing the EBS. The AFN serving as the centralized key management entity must store all $k + m$ keys, and each MSN must store k keys. Note that the key subset held by each MSN is unique. We illustrate an instance of EBS(10,3,2) in Table 1. The i -th ($1 \leq i \leq 10$) node in the cluster is denoted as N_i , and the j -th ($1 \leq j \leq 5$) administrative key is denoted as Ka_j . An entry marked with a “1” indicates that the node in the corresponding column possesses the administrative key of the corresponding row.

These administrative keys serve as key encryption keys. They allow the AFN to establish, refresh, and revoke any session or group key belonging to any node. The basic scheme presented above is efficient, and functions well under ideal conditions. However, this scheme has a significant drawback—it is not resilient against node

captures. In the next section, we present the threat model that was considered when designing SECK.

Table 1. Sample administrative key sets using EBS(10,3,2)

	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}
Ka_1	1	1	1	1	1	1	0	0	0	0
Ka_2	1	1	1	0	0	0	1	1	1	0
Ka_3	1	0	0	1	1	0	1	1	0	1
Ka_4	0	1	0	1	0	1	1	0	1	1
Ka_5	0	0	1	0	1	1	0	1	1	1

4. THE THREAT MODEL

We consider an attack scenario where an adversary is able to compromise one or more nodes of a WSN. Specifically, we consider three different cases with differing degrees of severity. Researchers have pointed out that there really is no sure and efficient way to readily detect a node capture [3,7]. Because of this, it is important to emphasize the network’s survivability given a number of undetected node captures, in addition to the ability to recover from a single node capture at a time.

The attack scenario that possesses the greatest threat to the bottom tier of the network is the compromise of an AFN. As a result of an AFN capture, all data collected by MSNs in that cluster is compromised. After the detection of the AFN capture, the following steps need to be executed to restore normal operations of the cluster: (1) notify MSNs of the capture, (2) establish a new AFN for each MSN, and (3) establish a new security relationship between the MSN and the AFN in the second step. If this re-clustering is not supported by the network, all MSNs within the affected cluster are considered off-line until a new AFN is deployed. The AFN that is captured contains a full set of administrative keys that will need re-keying. In order to localize the necessary re-keying operations, it is necessary for administrative keys to be independently replaced. If the administrative key sets were globally used, all keys for all clusters would be compromised as a result of a single AFN capture.

The next threat is the compromise of MSNs within the same cluster. In the basic scheme described in Section 3, it is possible for an adversary to capture all the administrative keys of a cluster with only a few MSN captures. This is possible if the adversary is able to pick the nodes in a strategic manner; in the running example of Section 3, the strategic choice would be to compromise N_1 and N_6 , which would reveal all five administrative keys. Therefore, if capture detection is not possible, or not prompt, the entire cluster will likely be rendered insecure unless a method of recovery is developed. SECK provides a recovery method to salvage the remaining uncompromised nodes within the cluster.

The least severe physical threat that an attacker may pose is to compromise nodes randomly throughout the network. The major security benefit of a clustered

architecture is its ability to localize the effects of attacks on randomly chosen nodes. A secondary benefit is that multiple decentralized attacks have no increased effect. If two adversaries located randomly throughout the network compromise a node each, combining the information obtained from these nodes provides no added benefit, assuming the nodes are not within the same cluster.

5. THE COMPLETE SPECIFICATION OF SECK

We introduce the mechanisms needed to complete our key management scheme, SECK. We start our discussions by describing a location-training scheme that establishes the clusters, and administrative key identifiers used in SECK. Then, we present security mechanisms for replacing administrative keys compromised by MSN captures. Finally, we describe a secure re-clustering scheme for salvaging MSNs within a cluster after the AFN has been captured.

5.1. Location Training

The location-training scheme will establish a coordinate system and assign each MSN a cluster coordinate identifier, which establishes unique administrative key subsets. In addition, this scheme supports the ability to store a *next-hop* node location in the direction of an optimal backup AFN, AFN_b . The cluster coordinate established is $(tree, hopcount)$ where $tree$ is an integer assigned by each AFN, and $hopcount$ is the MSN's distance from the primary AFN, AFN_p . We define a *tree* as a set of MSNs that share the same tree root when routing data, where the tree root is an MSN that is an established one-hop neighbor of AFN_p . Our location-training scheme supports clusters with any number of hops.

It is assumed that at this point MSNs have completed neighborhood discovery, and every MSN and AFN is aware of all reachable one-hop neighbors through broadcasted 'hello' messages. Each AFN first broadcasts a list of one-hop neighbors that it has discovered. We denote S_{h1} as the set of one-hop neighbor nodes. Each entry in this broadcasted list is a tuple of the MSN ID and its assigned tree number (we will describe how AFNs assign tree numbers in Section 5.2). All one-hop nodes will serve as the tree root for any multi-hop nodes established in that tree.

$$AFN_i \Rightarrow N_1, \dots, N_m :$$

$$ID_{AFN_i} \parallel (ID_1, tree_1) \parallel (ID_2, tree_2) \parallel \dots \parallel (ID_{|S_{h1}|}, tree_{|S_{h1}|}),$$

where m denotes the number of MSNs within the transmitting range of the AFN. MSNs search this list for their ID and the ID of their discovered neighbors. If a node

finds its ID on this list, it assigns its cluster coordinate as $(tree_{ID}, 1)$ for its newly established AFN_p . If an MSN does not find its ID, but finds the ID of a neighbor, say i , it assigns itself its cluster coordinate $(tree_{ID_i}, 2)$ —the first entry corresponds to the tree number of the neighbor MSN, and the second entry indicates the hop count from AFN_p . A second hop MSN can figure out that it is 2 hops away from AFN_p by observing that its neighbor is a tree root. MSNs with multiple neighbors on AFN_p neighbor list should randomly choose which tree to join (this ensures a more uniform distribution of root node selection).

The group of second hop neighbors S_{h2} , initiate the propagation of the coordinate establishment message by broadcasting their cluster coordinate and ID_{AFN_p} as follows:

$$\forall N_i \in S_{h2}, N_i \Rightarrow \text{neighbor} :$$

$$ID_{AFN_p} \parallel (tree, hopcount)_{N_i}.$$

Upon hearing this message, MSNs not yet holding a primary cluster coordinate will know how many hops away from AFN_p they are, and what their cluster coordinate should be. As these messages propagate, they will establish the primary AFN and cluster coordinate for all MSNs. A MSN will always forward the first coordinate establishment message it receives. Once a MSN begins to receive additional coordinate establishment messages, it will forward this message only if it is the closest backup AFN yet to hear, that is, $hopcount_{new} < hopcount_{backup}$. We describe, in Section 5.4, the importance of establishing AFN_b .

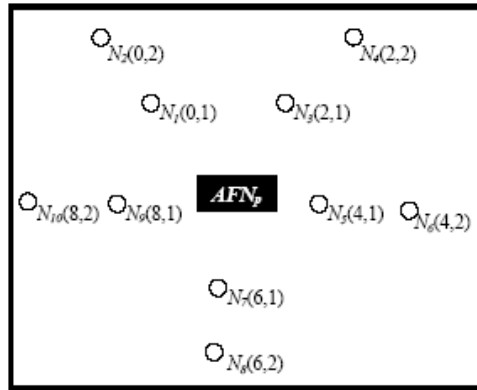


Figure 1. Optimal cluster coordinates established in a single cluster.

Every MSN, with the exception of the $|S_{h1}|$ established root nodes, must broadcast one message when establishing their primary cluster coordinate. Every MSN must transmit at least one additional message when establishing AFN_b . We will assume that the ideal is the case, and the optimal AFN_b is established with the first received coordinate

establishment message. The total communication overhead estimated for this process is $1 + 2 \cdot n - |S_{h1}|$ transmissions within each cluster. Using our running example, Figure 1 shows the optimal cluster coordinates established within a ten MSN cluster.

5.2. Administrative Key Establishment

The administrative key subsets are determined by the unique identifier found by summing the two elements of a node's cluster coordinate. Each AFN is then responsible for generating and distributing the tree administrative keys for each tree in its cluster. This requires one message for each of the unique key subset identifiers used in the cluster. The first step is to generate a tree administrative key for each tree in the cluster. We denote Kt_j as the tree administrative key for the j -th tree, and d as the number of trees in a cluster, that is $d = |S_{h1}|$. Recall that a *tree* consists of all nodes that utilize the same forwarding route. From Figure 1, we see that N_1 and N_2 are the members of $tree_0$. Each Kt_j is encrypted with the h different key subsets, S_i within that tree as follows:

AFN \rightarrow all MSNs in $tree_{(h,j)}, 0 \leq j < d$:

$$\mathbf{E}_{K_1}(\mathbf{E}_{K_2}(\dots \mathbf{E}_{K_{|S_i|}}(Kt_j)\dots)), (h \cdot j) < i \leq (h \cdot j + h),$$

where $\mathbf{E}_K(\cdot)$ denotes Encryption with key K , and $K_1, K_2, \dots, K_{|S_i|} \in S_i$.

5.3. Administrative Key Recovery

If all administrative keys have been compromised due to node capture within an isolated set of trees, then the remaining secure trees may survive by using their tree administrative key to reestablish their administrative key subsets.

In Section 4 we showed why the capture of a small number of MSNs can possibly compromise most or all of the administrative keys in the basic scheme. In such a scenario, even the MSNs that were not captured are excluded from any further communications with the rest of the network because all of the administrative keys of the cluster have been compromised, thus making it impossible to distribute new session keys to those MSNs. Suppose that an AFN receives notification that a set of nodes, which we denote as S_c , has been compromised, resulting in the compromise of all administrative keys. In response, the AFN computes the set of trees unaffected by S_c , denoted as S_{ut} , (i.e., the trees not containing any node from S_c).

$$S_{ut} = \{tree_i : tree_i \cap S_c = \emptyset, 0 \leq i < d\}.$$

The AFN then creates $|S_{ut}|$ messages, each containing a set of new administrative keys, and transmits the messages to the appropriate trees as shown in the following notation.

$\forall tree_i \in S_{ut}, AFN \Rightarrow tree_i$:

$$\mathbf{E}_{Kt_i}(\mathbf{E}_{Ka_1}(Ka_1') \parallel \mathbf{E}_{Ka_2}(Ka_2') \parallel \dots \parallel \mathbf{E}_{Ka_{k+m}}(Ka_{k+m}')).$$

5.4. Reactive Re-clustering After AFN Capture

Unlike an MSN, an AFN carries out several key tasks that are essential to its cluster. Because the loss or capture of an AFN can incapacitate the entire cluster, giving the MSNs the ability to recover from such a situation greatly improves the survivability of the cluster by removing the single point of failure [8]. To keep the MSNs operational after an AFN capture, we need to merge MSNs into neighboring clusters and establish a new secure relationship with a backup AFN.

We utilize the preloaded base station pairwise keys, Kp_i , to allow the base station to facilitate a security relationship between a recovering MSN, N_i , and AFN_b .

Upon detection that N_i 's primary AFN has been captured, a MSN constructs a short recovery request message, destined for the base station (via *next-hop* and AFN_b). N_i sends its node identifier, ID_{N_i} and a MAC, generated with its base station pairwise key Kp_i .

$$N_i \rightarrow next-hop : ID_{N_i} \parallel ID_{AFN_b} \parallel nonce_i \parallel MAC_{Kp_i},$$

where MAC_{Kp_i} represents N_i 's MAC. We assume that this *next-hop* node would route this message to AFN_b in the same manner as forwarding sensed data to AFN_b for data aggregation. When this message reaches AFN_b , this AFN creates a signature of this message, appends its signature to the message, and forwards the message to the base station.

$AFN_b \rightarrow$ base station :

$$ID_{N_i} \parallel ID_{AFN_b} \parallel nonce_i \parallel MAC_{Kp_i} \parallel Sign_{AFN_b},$$

where $Sign_{AFN_b}$ represents AFN_b 's signature. When the base station receives this message, it verifies the identity of the MSN using the original MAC, and then authenticates the AFN based on its signature. If both nodes are authenticated, the base station establishes a shared secret between the AFN_b and N_i by returning two instances of a new key, $K_{AFN_b-N_i}$.

base station $\rightarrow AFN_b$:

$$ID_{N_i} \parallel ID_{AFN_b} \parallel \mathbf{E}_{K_{AFN_b-N_i}}(K_{AFN_b-N_i} \parallel ID_{N_i} \parallel ID_{AFN_b} \parallel nonce_i),$$

base station $\rightarrow N_i$:

$$ID_{N_i} \parallel ID_{AFN_b} \parallel \mathbf{E}_{Kp_i}(K_{AFN_b-N_i} \parallel ID_{N_i} \parallel ID_{AFN_b} \parallel nonce_i),$$

where K_{AFN_b} represents the public key of AFN_b . Based on the successful receipt of the encrypted key, the MSN and backup AFN can be assured of the other's authenticity. Using the newly established $K_{AFN_b-N_i}$, AFN_b finalizes N_i 's membership in its cluster by distributing a new subset of administrative keys and its subset identifier.

6. PERFORMANCE EVALUATION

We evaluate the communication, storage, and security overhead of our clustered key management scheme, SECK. We highlight the benefits of our scheme as they compare to other WSN keying schemes. We perform the security analysis given in Section 6.3 is based on the threats identified in section 4.

6.1. Communication Overhead

The communication overhead of our scheme is incurred during the initial key establishment phase and during periodic key maintenance procedures.

SECK requires a location-training process to establish administrative key sets and a distribution process to distribute a group key. The total number of transmissions required per cluster for our location-training process is $1 + 2n - d$. For group key distribution and re-keying, a total of $k + m$ messages need to be transmitted within the cluster for each key.

Because the establishment overhead for SECK is reliant on the size of the network, it performs better for smaller clusters. We assume a typical cluster size of approximately 50 nodes. Because our architecture is modular, we are able to keep cluster sizes small, and thus are able to minimize the number of hops that need to be traveled when transporting group keys. This characteristic becomes increasingly important as the network size increases.

6.2. Storage Overhead.

Initially, our scheme requires the complete administrative key set to be stored at each node. The complete administrative key set requires $128 \cdot (k + m)$ bits (here, we assume that AES is used as the encryption scheme). Additionally, one 128-bit base station pairwise key is stored at each MSN for a total initial storage requirement of $128 \cdot (k + m + 1)$ bits. Using a typical cluster sized example of EBS(56, 3, 5), each MSN would hold 144 bytes of initial keying data.

We now switch our attention to a MSN, post-deployment. At this point each MSN has deleted its unused administrative keys. The MSN has also been assigned one 128-bit tree administrative key. Using the same EBS(56, 3,

5), the MSN would store 80 bytes of keying data, or five 128-bit keys.

6.3. Robustness Against Node Captures

Node captures in hostile environments are inevitable, and a key management scheme should be able to recover from such attacks to be effective. Using the threats identified in Section 4, we show how robust SECK is against those attacks.

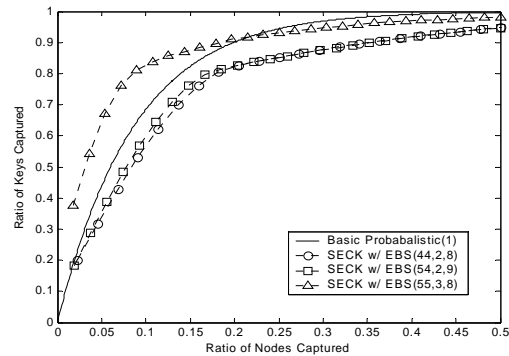


Figure 2a. Ratio of keys captured vs. ratio of nodes captured.

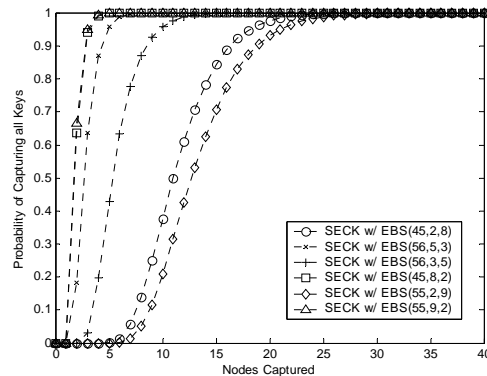


Figure 2b. Probability of capturing all keys vs. number of nodes captured.

The plots in Figure 2a indicate that SECK's resiliency against node (MSN) captures is comparable to that of the basic probabilistic pairwise scheme of Eschenauer and Gligor [2]. This demonstrates SECK's good resiliency property, as the Eschenauer and Gligor scheme is widely regarded as resilient.

From Figure 2b, in the best case, once 14 MSNs have been randomly captured, there is a good (> 60%) chance that the complete administrative key set has been compromised. If all the administrative keys of a cluster are compromised, the MSN administrative key recovery procedure is necessary. We now show the effectiveness of the administrative key recovery procedure for the best and worst case scenarios assuming that 14 MSNs have been captured.

The worst case occurs when each of the 14 captures was

from a unique tree. This leaves $d - 14$ trees unaffected, and $(d - 14) \cdot h$ nodes recoverable. If we approximate d with n/h , the ratio of nodes that can be recovered is $(n - 14h)/(n - 14)$. Suppose that every MSN in the cluster is within 2 hops of the AFN (i.e., $h = 2$) and EBS(55, 2, 9) is used. Then our procedure recovers 0.66 of the uncompromised nodes.

The best case occurs when the attack is completely localized. That is, all nodes within a single tree are captured before the attacker moves on to the next tree. This will affect $\lceil 14/h \rceil$ trees, leaving $d - \lceil 14/h \rceil$ trees unaffected. If we again approximate d with n/h , the ratio of nodes that can be recovered is $(n - 14)/(n - 14) = 1$. From the above analyses, we can observe that SECK's recovery procedure performs best in localized attacks.

7. RELATED WORK

In the past few years, several technical approaches have been proposed to provide WSNs with the required confidentiality and authentication services via pairwise secret (or key) sharing [2,7,10,11]. Eschenauer and Gligor. [2] propose a probabilistic distribution of pairwise keys, randomly chosen from a global key pool. Chan et al. [10] further extended this idea to provide localized attack resiliency. Although pairwise keys do need to be established between nodes, most WSN applications require additional levels of key sharing—specifically, group keys.

Carmen et al. [9] conducted a comprehensive analysis of various group key schemes. The authors conclude that the group size is the primary factor that should be considered when choosing a scheme for generating and distributing group keys in a WSN. Zhu et al. [7] propose a comprehensive key management scheme called LEAP that establishes multiple keys for supporting neighborhood as well as global information sharing. Although LEAP includes several promising ideas, it does not adequately address scalability issues concerning the distribution and maintenance of group keys.

To address the difficult problem of scalability, many have proposed hierarchical network architectures, similar to the one described in this paper. In [3,4,6], authors utilize a clustered and hierarchical network architecture for key management. Jolly et al. [3] employ a hierarchical network organization to establish gateway-to-sensor keys. The clustering technique used by Jolly et al. was originally developed by Gupta and Younis [12]. Gupta and Younis propose to form a cluster in which all nodes are within one hop of the cluster head by using GPS signals. Eltoweissy et al. [4,6] present another hierarchical key management scheme based on the Exclusion Basis System to efficiently maintain group and session key information. This approach supports key recovery only if node captures can be

immediately detected.

Bohge and Trappe [1] propose a hierarchical authentication technique to establish and recover keys. Their approach requires a broadcast authentication scheme and for this purpose, they employ a variation of μ TESLA [13].

8. CONCLUSION

In this paper, we have proposed a novel key management scheme. SECK is a key management solution that includes: (1) a location training scheme for establishing a cluster, and cluster coordinate system used in the MSN recovery procedure; (2) a scheme for establishing and recovering administrative keys; (3) a scheme for distributing session keys using administrative keys; and (4) a procedure for salvaging MSNs in the event that their AFN has been captured. Through analytical and simulation results, we have shown that SECK is resilient to node and key captures while maintaining low levels of communication and computation overhead.

REFERENCES

- [1] M. Bohge, W. Trappe "An authentication framework for hierarchical ad hoc MSN networks," in *Proc. of the ACM workshop on Wireless security* pp.79–87, 2003.
- [2] L. Eschenauer and V.D. Gligor. "A key-management scheme for distributed MSN networks," In *Proc. Of the 9th ACM Conf. on Computer and Communications Security*, pp. 41–47, November 2002.
- [3] G. Jolly, M Kusu, P Kokate, "A Hierarchical Key Management Method for Low-Energy Wireless MSN Networks," In *Proc. Eighth IEEE Symposium on Computers and Communication*, 2003. pp. 335-340.
- [4] M. Eltoweissy, A. Wadaa, S. Olariu, and L. Wilson "Scalable cryptographic key management in wireless sensor networks," in *Journal of Ad Hoc Networks*. November 2004.
- [5] M. Eltoweissy, H. Heydari, L. Morales, and H. Sudborough, "Combinatorial Optimizations of Group Key Management," *Journal of Networks and Systems Management*, March 2004. pp. 30–50.
- [6] M. Eltoweissy, M. Younis, K. Ghumman, "Lightweight Key Management for Wireless MSN Networks," In *Proc. IEEE International Conference on Performance, Computing, and Communications*, April 2004. pp. 813–818.
- [7] S. Zhu, S. Setia, S. Jajodia, "LEAP: Efficient Security Mechanisms for Large-Scale Distributed MSN Networks," In *Proc. of ACM conference on Computer and communication security*, pp. 62–72, 2003.
- [8] D. Carman, P. Kruus, B. Matt. "Constraints and approaches for distributed MSN network security," *NAI Labs Technical Report No. 00010*. 2000.
- [9] D. Carman, B. Matt and G. Cirincione. "Energy-efficient and Low-latency Key Management for MSN Networks." In *Proc. of 23rd Army Science Conference*. Dec 2-5 2002 Orlando Florida.
- [10] H. Chan, A. Perrig, and D. Song. "Random key predistribution schemes for MSN networks" In *Proc. IEEE Symposium on Security and Privacy*, pp. 197–213, California, May 2003.
- [11] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," In *Proc. ACM Conference on Computer and Communications Security (CCS '03)*, pp. 52–61, 2003.
- [12] G. Gupta, M. Younis, "Performance Evaluation of Load-Balanced Clustering of Wireless MSN Networks," In *Proc. 10th International Conference on Telecommunications*, February 2003. pp. 1577–1581.
- [13] A. Perrig, et al., "SPINS: Security protocols for sensor networks," *Wireless Networks*, Vol. 8, No. 5, pp.521–534, 2004.