ISECK: Improved Survivable and Efficient Clustered Keying for the Application of Hydraulic Pressure Distribution for Pipeline Networks by Wireless Sensor Networks

Reza Aminzadeh, Feraydune Kashefi
Khavaran Higher Education Institute
{reza.aminzadeh, fred.kashefi}@ieee.org
Paper Refrence Number: 52123-400
Presenter: Reza Aminzadeh

Abstract

In this paper we propose a key management scheme for an application of wireless sensor networks. Our application consists of a number of clustered sensor nodes that measure the pressure in water pipelines. Sensor nodes communication is based on the IEEE802.15.4 standard by the help of Zigbee modules. The data is collected in 15-minute period intervals and is sent via GPRS to the control center. The main purpose of this research is to identify the unbalanced water pressure distribution problem in cities. We chose the key management scheme to be the suitable solution for our application. As our application with its large scale sensor network deployment and layout, is in a sensitive and vulnerable to the intruders in a hostile environment, we introduced and tested our system with a different disruptions scenario and based on the collected result we came up with the ISECK security package in order to have a strong resilience network against node captures. ISECK is an improved version of survivable and efficient clustered keying protocol. We also considered the energy efficient protocol. According to the test results and simulation we show that our protocol is more energy efficient and stronger than the previoulsly proposed SECK.

Key Words: Hostile environment, Hydraulic Pipelines, ISECK, Key Management, Node Capture.

1. Introduction

Key management is crucial to the secure operation of wireless sensor networks (WSNs). A large number of keys must be managed in order to encrypt and authenticate all sensitive data. Large scale WSNs are highly vulnerable to attacks because of their numerous miniaturized resource-constraint devices, closely interaction with the physical environment, and wireless communication links. The objective of key management is to dynamically establish and maintain secure channels among communicating parties [2]. In our previous work [1] we described a new application of wireless sensor networks for real time monitoring of water pressure distribution in urban areas. The objective of this application is to create a continuous monitoring system to enable continuous monitoring for municiples water pipeline systems. This system also employs the content model for understanding the status of water pressure in the pipeline network. We implemented our system in different areas of Mashhad metropolis. In such a network some security requirements should be emphasized such as resilience against node capture, replication, revocation and participation. Scalibility is an important factor to design a suitable key management protocol for this application. In this project we also have an energy-efficient key management algorithm used for the purpose of staying aline with the world energy conservation. According to the received test results the stability of our system is maintained and the level of security against node capture is higher than the previously

presented algorithms while the computational processes and energy consumption are at a low rate.

2. Our Method

In this project we propose (ISECK) which is an improved version of Survivable and Efficient Clustered Keying (SECK). SECK is a dynamic key management scheme that is appropriate for a network with a hierarchical architecture. It was originally proposed by Chorzempa et al. [2], which is a self-organizaing scheme. By using ISECK we have a reliable sensor network that can be implemented in hostile environments. Using analytical and simulation results, we show that ISECK is robust against the attacks that we identify in the threat model we will describe in this section. Moreover, our results show that ISECK incurs low communications and storage overhead on the sensor nodes. The system architecture was designed in a hierarchical structure because it can improve a network's robustness against node or key captures by limiting the effects of an attack to a certain portion of the network. We designed our application into a clustered architecture. In this architecture each cluster has aggregation and forwarding nodes (AFNs) that are equipped with high-end embedded processors. Each AFN has the capability of communicating with other AFNs over long distances. Another part of each cluster is micro-sensor nodes (MSNs) which consist of low-end battery powered sensor nodes for short range communications at low data rates. In other words MSNs are sensor nodes that measure water pressure in pipelines and AFNs act as coordinators in each cluster. In a cluster, each MSN sense the water pressure and forwards it to the AFN. The AFN aggregate the forwarded data that comes from MSNs in that cluster and then sends the collected data to the next hop AFN towards the Base Station (BS) which is the ultimate destination for data streams from all the AFNs. The BS has powerful data processing capabilities and is directly connected to an outside network. Within each cluster, a set of keys must be deployed and managed to secure communications between the MSNs and the AFN. Figure 1 shows the physical architecture of our application. We designed relay nodes for underground to above communication in our previous work [1].

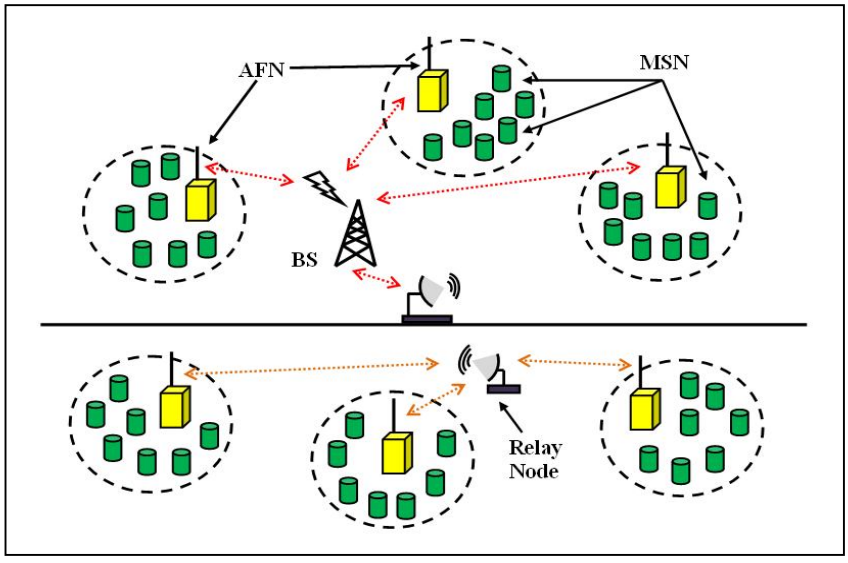

Fig.1 Physical Architecture of the proposed application.

1. Fundamental design of ISECK

To distribute and refresh session keys, ISECK assigns a set of administrative keys to nodes in a sensor network. Managing keys are done by Exclusion Base System (EBS). EBS is based on a combinatorial formulation of the group key management problem. It essentially provides a mechanism for establishing the administrative keys held by each node. EBS function is defined as EBS (n, k, m) where *n* is the number of nodes 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. Each MSN stores *k* keys. We denote the administrative key set for all nodes in the system as *U*, where

$$|U| = k + m \tag{1}$$

For full details of the EBS, please see [3, 4]. The AFN serving as the key management entity for its cluster must store all $k + m$ keys, and each MSN must store k keys. Note that the key subset held by each MSN is unique. This feature is utilized by the AFN to distribute session keys, that is, keys of this subset are used to encrypt the session keys before distribution. We illustrate an instance of EBS (16, 3, 3) in Table 1. An entry marked with a "1" indicates that the node in the corresponding column possesses the administrative key of the corresponding row. To initially distribute a cluster-wide key to all members of the cluster, one message is broadcasted by the AFN to all members of the cluster for each administrative key. This requires $k+m$ short broadcasts by the AFN.

	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	N16
Ka1	1	1	1	1	1	1	0	0	0	0	0	0	1	1	0	0
Ka2	1	1	1	1	0	0	0	0	0	1	0	1	0	0	1	1
Ka3	1	0	0	0	1	0	0	1	1	1	1	1	1	0	0	1
Ka4	0	1	0	0	0	0	1	1	1	0	0	1	0	1	1	0
Ka5	0	0	1	0	0	1	1	0	1	0	1	0	1	0	0	1
Ka6	0	0	0	1	1	1	1	1	0	1	1	0	0	1	1	0

Table1. Sample Administrative Key Subset Using EBS (16, 3, 3).

2. The Threat Model

The first scenario requires an adversary to locate and visually distinguish an AFN from a MSN. Then an adversary must extract the sensitive contents of the AFN (e.g., keys). If an AFN capture is not immediately detected, all data collected by MSNs in that cluster will be compromised. The AFN that is captured contains a full set of administrative keys that will need re-keying. If reclustering is not supported by the network, all MSNs within the affected cluster are considered off-line until a new AFN is deployed. To localize the necessary re-keying operations, it is necessary for administrative keys to be independently replaced. If the administrative key sets were globally calculated and distributed to all AFNs, all keys for all clusters would be compromised as a result of a single AFN capture. In the second attack scenario, MSNs within the same cluster are compromised. ISECK provides a recovery method to salvage uncompromised MSNs within a cluster when some or even all administrative keys are compromised. In the third attack scenario, an attacker may compromise nodes randomly throughout the network. An advantage of our system is that multiple decentralized attacks may not have increased the effect compared to a single attack instance. If two adversaries located randomly throughout the network compromising one node each, combining the information obtained from these nodes provides no added benefit, assuming the nodes are not within the same cluster. Of course, in our system, each AFN generates its administrative keys independently from others.

3. Required Keys

Initially each MSN will be deployed with the complete administrative key set. After a short setup period, only a subset of these keys will be stored. In addition, each MSN will be required to store key *Kpi*, which is a pair-wise secret key shared with the BS, and one tree administrative key *Kti*. *Kpi* is needed for the reclustering process after the capture of the AFN has been detected. The key *Kti* is used to replace compromised administrative keys after MSN captures have been detected.

4. Location Training

This scheme enables an MSN to store the location of the next-hop MSN, *Nx*, in the direction of its backup AFN, *AFNb*. Each MSN is absorbed into *AFNb*'s cluster in the event of *AFNp*'s compromise or failure. A cluster coordinate established is given as (*tree*, *hopcount*), where *tree* is an integer assigned by *AFNp*, and *hopcount* is the MSN's distance from *AFNp*. We define a *tree* as a set of MSNs that routes packets through the same tree root when forwarding data to the *AFNp*, where the tree root is an MSN that is one hop away from the *AFNp*. Now, each AFN broadcasts the list of one-hop neighbors that it has discovered to all MSNs within each transmission range. 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 (*treeID*, 1) and becomes one of the tree roots of *AFNp*. If an MSN does not find its ID, but finds the ID of a neighbor, say *i*, it assigns itself the cluster coordinate (*treeIDi*, 2) *treeID* represents the tree number of the neighbor MSN and the second entry indicates the hop count from *AFNp*. MSNs with multiple neighbors on the neighbor list of *AFNp* should randomly choose which tree to join. An MSN will always forward the first coordinate establishment message it receives. Every MSN must transmit at least one additional message when establishing *AFNb*. Using our running example, Figure 2 shows cluster coordinates established within a ten MSN cluster. This figure shows the optimal case, because each node receives a unique cluster coordinate.

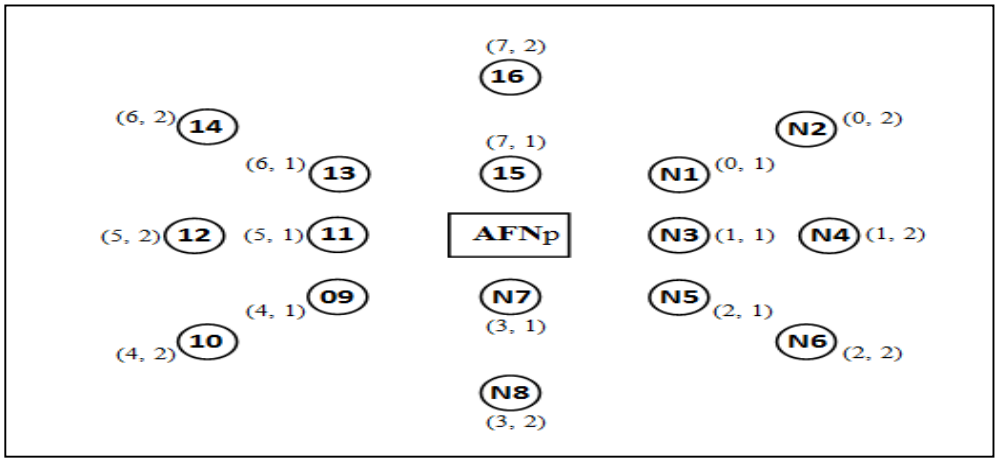

Fig.2 Optimal Cluster Coordinates Established In a Single Cluster.

5. Establishment of Administrative Keys

We assume that the expected number of hops can be estimated based on the density of AFN deployment relative to MSN deployment. We denote the number of neighbors found by an AFN as *d*, and the maximum number of estimated hops within a cluster as *h*. We assign tree numbers as *tree* $i = d \cdot h$. Each unique cluster will generate a unique key subset identifier in the range of [1, $d \cdot h$]. Each AFN is responsible for generating and

distributing the tree administrative keys for each tree in its cluster. This requires $d \cdot h$ messages transmitted by the AFN. It is important for the administrative key set, *U* to be independently updated within each cluster after all tree administrative keys have been established. If the same set of administrative keys is maintained globally, the compromise of a single cluster's keys would compromise the entire network's keys. In order to isolate an attack, the administrative keys must be independently generated in each cluster and not shared among clusters.

6. Administrative Key Recovery

When all of the administrative keys of the cluster have been compromised, even the MSNs that are not captured are excluded from any further communications with the rest of the network. An AFN at some point receives notification that a set of nodes, which we denote as *Cn*, has been compromised, resulting in the compromise of all administrative keys. In response, the AFN computes the set of trees not containing any node from *Cn*, which we denote as *Nt*. The AFN then creates |*Nt*| messages, each containing a set of new administrative keys, and transmits the messages to the appropriate trees. This technique cannot salvage MSNs that belong to a tree in which some of its nodes have been compromised.

7. MSN Addition

MSNs are randomly deployed and each new MSN will contain a unique base station pair-wise key, *Kpi*, which allows the BS to facilitate a new security relationship between the new MSN and existing AFN. The only difference between deploying a new MSN and reclustering an MSN is that the newly deployed MSN has no knowledge of a backup AFN. The newly added MSN broadcasts a short "hello" message to announce its presence. Each neighbor that overhears this message replies with its cluster information. *Nnew* uses these replies to determine the most appropriate cluster to join and the most efficient neighbor to use as a next-hop node to that cluster's AFN. *Nnew* determines the most appropriate AFN. The neighbor with the smallest *hopcount* is the most efficient node to use as a next-hop node in the direction of the detected *AFN*. Figure 3 depicts our method's full diagram.

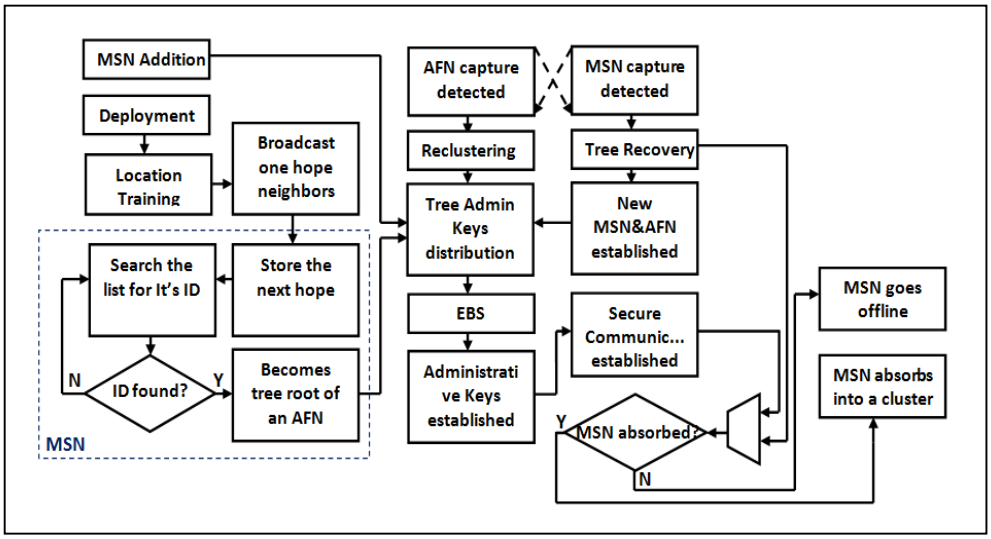

Fig.3 ISECK full function diagram.

3. Simulation and Analysis

1. *Robustness against MSN Node Capture*

In the example given in Table 1, it is possible for an adversary to capture all the administrative keys of a cluster with only a small number of node captures by strategically selecting the nodes to capture. Fortunately, such attacks are difficult to carry out; to be successful, an adversary needs to know which administrative keys are stored in each MSN. Of course, the adversary can always randomly choose the nodes to capture and hope that a large proportion of administrative keys are revealed by those captures. Using simulations, we show that when the nodes are compromised randomly, then the proportion of compromised keys is commensurate with the wellknown probabilistic key distribution scheme proposed in Eltoweissy and Gligor [5]. Figure 4 shows that ISECK's robustness against random node (MSN) captures. Now we attempt to answer the following question: how many MSNs must be captured before all the administrative keys of a cluster ($k + m$ keys in total) are compromised? We are interested in instances of ISECK that support about 50 nodes, as that is the expected size of a typical cluster that we envision. The figure shows that the ratio k/m has a direct impact on the robustness of ISECK against node captures. Decreasing the ratio improves robustness against node captures and increasing the ratio has the opposite effect. However, setting the ratio k/m to a low value incurs a cost. As the ratio decreases, the communication overhead required to distribute session keys increases. One can see that there exists a communication overhead versus node capture resiliency trade-off.

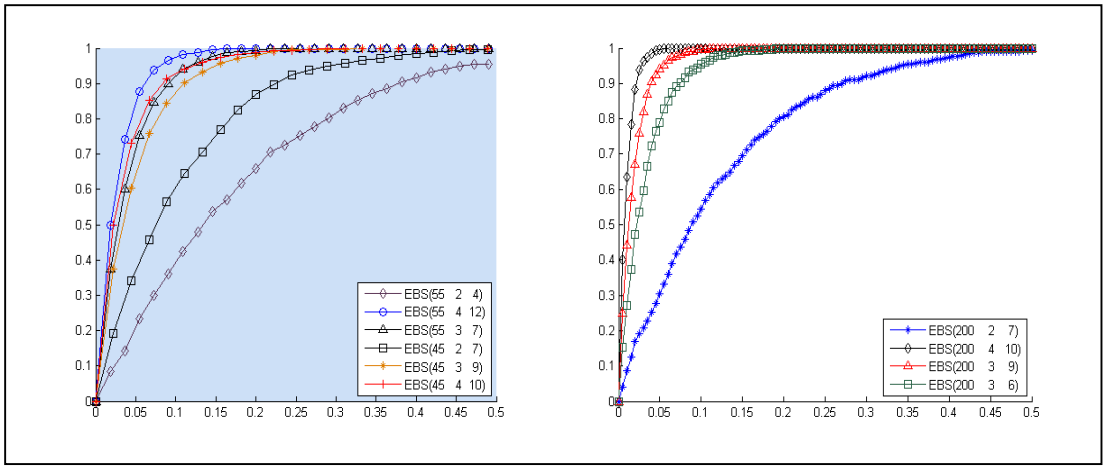

Fig.4 Expected Ratio of Keys Captured vs Ratio of Nodes Captured.
(Ratio of nodes captured: horizontal axis, Ratio of keys captured: vertical axis)

2. Administrative Key Recovery Procedure

If 16 MSNs have been randomly captured in this case, there is a good chance (>60 percent) that the complete administrative key set has been compromised. If all the administrative keys of a cluster are compromised, the network needs to execute the MSN administrative key recovery procedure. In the following paragraphs we discuss the effectiveness of the administrative key recovery procedure in two distinct cases; best and worst case scenarios. Assuming that x MSNs have been captured. The worst case occurs when each of the x captures occurs in separate trees. This leaves $d - x$ trees unaffected, and $(d - x) \cdot h$ nodes can be recovered. If we approximate d with n/h, the ratio of nodes that can be recovered is $(n - x \cdot h)/(n - x)$. Suppose that every MSN in the cluster is within two hops of the AFN (i.e. $h = 2$) and EBS (55, 4, 12) is used. Then our procedure recovers

0.63 of the uncompromised nodes in the worst case. 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 $[x/h]$ trees, leaving $d - [x/h]$ trees unaffected. If we again approximate d with n/h, the ratio of nodes that can be recovered is $(n-h \cdot [x/h])/(n-x) \approx 1$. From the above analysis we can observe that ISECK's recovery procedure performs best in localized attacks. In Figure 5, we have plotted the ratio of recoverable nodes in the best and worst case attack scenarios. In this figure, we have highlighted the case where 16 nodes have been captured. In practice, the actual recovery ratio is expected to be somewhere between 0.6 and 1.0 when 16 nodes are captured. Also it can be seen in figure 5 (right) that as the number of nodes increases, the ratio of key recovery increases too.

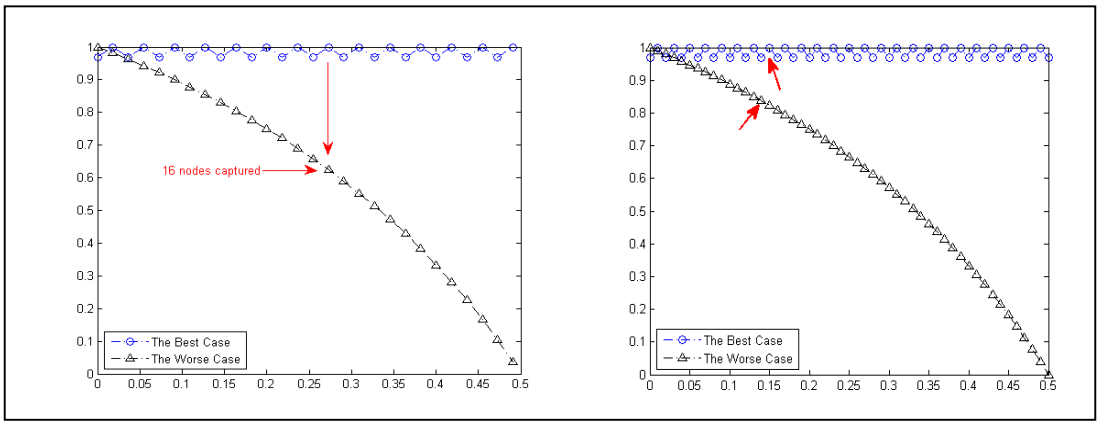

Fig.5 Administrative key Recovery Procedure Simulation.
(Ratio of nodes captured: horizontal axis, Ratio of remaining nodes recovered: vertical axis)

3. Storage overhead

Prior to deployment, each MSN needs to store a key subset matrix and the complete administrative key set. The key subset matrix is a $(k+m) \times n$ bit matrix that identifies the keys associated with each subset. Table 1 is a sample 6×16 key subset matrix. The key subset matrix requires $n \cdot (k+m)$ bits to store, and the complete administrative key set requires $128 \cdot (k+m)$ bits to store (here, we assume that 128-bit AES keys are used). Additionally, one 128-bit base station pair-wise key is stored at each MSN, for a total initial storage requirement of $((128 + n) \cdot (k + m) + 128)$ bits. With this formula, one can calculate that each MSN would need to store 304 bytes of initial keying material if EBS (55,4,12) is employed. After deployment, each MSN removes most of its initial keying material immediately after the conclusion of the location training processes.

4. Comparison of Communication Overhead

A unique feature of ISECK is its use of both administrative keys and session keys. Because of this feature, we cannot compare ISECK, in its entirety, with a single scheme. In ISECK, every MSN establishes a pair-wise key only with its primary AFN. In SECK, each node calculates a pair-wise key to communicate with the BS. This calculated pair-wise key is unicasted to each neighbors of the node d. To compute the energy dissipation incurred by ISECK, we assume that ISECK has the same message format as that used in SECK for key distribution. Here, we only consider communication-related energy dissipation. In Figure 6 we compare ISECK and SECK by plotting the energy dissipated by the network for key establishment as a function of the cluster size. We now switch our

attention to the communication overhead incurred by ISECK to maintain session keys. Our emphasis here is on the efficiency of session key distribution over multiple key update periods. For a network deployed in a hostile environment, where node captures are expected, it is important to be able to continuously distribute new session keys efficiently. In Figure 7 we compare ISECK and SECK in terms of the communication energy dissipated by the network due to session key redistributions.

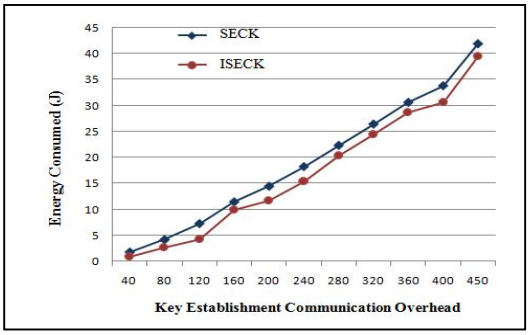

Fig.6 Key Establishment Energy overhead.

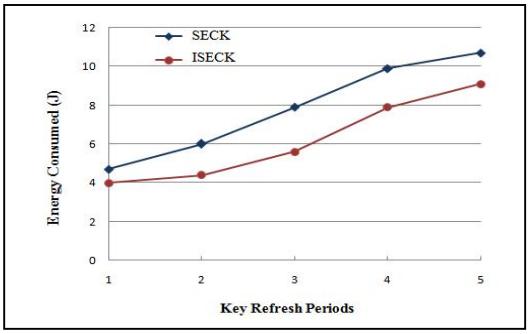

Fig.7 Key Refresh Communication overhead.

4. Conclusion

In a large-scale WSN deployed in a hostile environment, a key management scheme is the attractive monitoring management system of a large number key network. In this paper we described an improved form of cluster-based dynamic key management scheme that is designed to address the specific issue of municipal water pipeline pressure distribution. ISECK meets the stringent efficiency and security requirements of WSNs using a set of administrative keys to manage other types of keys such as session keys. ISECK is a key management solution for our application [1] that includes a location training scheme that establishes clusters and the cluster coordinate system used in the MSN recovery procedure; a scheme for establishing and updating administrative keys; a scheme for distributing session keys using administrative keys; a scheme for recovering from multiple node captures; and a scheme for clustering and salvaging MSNs in the event that their AFN has been captured. Through analytical and simulation results, we have shown that ISECK is resilient to node and key captures while carrying low level communication overhead in comparison to SECK.

5. References

[1] R. Aminzadeh, F. Kashefi, H. Alee, “Hydraulic pressure distribution for pipeline networks by wireless sensor networks”, *in Proc. 2010 IEEE Asia-Pacific conference on Applied Electromagnetics, APACE 2010, Malaysia, Nov 2010.*

[2] M. Chorzempa, J. Park, and M. Eltoweissy, “SECK: Survivable and efficient clustered keying for wireless sensor networks,” in *Proc. IEEE Workshop on Information Assurance in Wireless Sensor Networks*, pp. 453–458. Phoenix AZ, April 2005.

[3] M. Eltoweissy, A. Wadaa, S. Olariu, and L. Wilson, “Scalable cryptographic key management in wireless sensor networks,” *Journal of Ad Hoc Networks: Special issue on Data Communications and Topology Control in Ad Hoc Networks,* Vol. 3, No. 5, September 2005.

[4] M. Eltoweissy, H. Heydari, L. Morales, and H. Sudborough, “Combinatorial optimizations of group key management,” *Journal of Networks and Systems Management,* Vol. 12, No. 1, pp. 30–50, March 2004.

[5] 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 (CCS),* pp. 41–47. Washington D.C., November 2002.