2012 International Conference on Information and Network Technology (ICINT 2012)
IPCSIT vol. 37 (2012) © (2012) IACSIT Press, Singapore

An Efficient Key Distribution Scheme for Wireless Sensor Networks using polynomial based schemes

Sanjay Kumar [1 +], Deepti Dohare [2] and Mahesh Kumar [3]

[1] Benpour Technologies Pvt. Ltd, Dehradun,
[2] Citrix Systems, Bangalore
[3] ICFAI University, Dehradun

Abstract. Sensor Networks are highly vulnerable to attacks because, it consists of various resource-constrained devices and they communicate via wireless links. Establishment of pairwise keys between sensor nodes is used to realize many of the security services for Sensor Networks. Hence securely distributing keys among sensor nodes is a fundamental challenge in providing security services. In literature, there are two types of key distribution schemes: probabilistic and deterministic. Probabilistic schemes provide good resilience but the effect of a node capture spreads throughout the network and hence connectivity is affected. While deterministic schemes guarantee the key establishment between any pair of nodes in a network, they are less resilient to node capture.
In this paper, we proposed a key distribution scheme based on polynomials that provide full connectivity and very good resilience to node capture. It also has low communication overhead and very less space overhead. We have also proposed a novel method for the addition of new nodes in the network.

Keywords: Key Distribution schemes, Sensor Networks, Polynomials

1. Introduction

Sensor Networks have potential to provide economical solutions to many problems of practical importance. Some of the applications where Sensor Networks can be used are: Emergency Response System, Energy Management, Battlefield Management, Health Monitoring, Logistics and Inventory management etc. Sensor Networks are deployed in hostile environments. Environmental conditions along with resource-constraints give rise to many security threats or attacks. Adversary can physically capture and get the information contained in the sensor node, eavesdrop and inject new messages, modify messages, listen and analyse the messages to obtain the information contained in a message etc. Since solution to physical capture of a node is not possible, we can provide solutions to other security attacks. To defend against false data injection, authenticity of the sender must be checked so that sensors will not listen to unauthorized nodes. Modification of a message is detected by checking integrity of the message. To ensure confidentiality, the information contained in the message should not be displayed to any node other than sender and receiver. The message is sent encrypted with a key that is shared by sender and receiver. Keys play a central role in realizing security services like: authenticity, integrity, confidentiality etc. Keys need to be distributed securely among sensor nodes. For the distribution of keys, many key management schemes have been proposed in literature.

Basically there are two types of approaches for distributing keys among sensor nodes viz. probabilistic approaches [1, 2] and deterministic approaches [3–8]. Eschenauer and Gligor [1] proposed a probabilistic key-predistribution scheme for pairwise key establishment. For each sensor node, a set of keys are chosen from a big pool of keys and given to each node before deployment. Chan et al. [2] further extended this idea

[+] Sanjay Kumar. Tel.: + 91 9412148830.
E-mail address: sanjay_kumar@benpour.com.

and developed two key predistribution techniques: qcomposite key predistribution and random pairwise scheme[2]. Thus major drawback of probabilistic key distribution scheme is that, small number of compromised nodes affects a large number of nodes in the remaining network. Deterministic key distribution schemes have the advantage that the graph is fully connected because every node in the network can establish a key with any other node. Deterministic schemes guarantee the establishment of a unique key between any pair of sensor nodes in the network. When nodes are captured, full network connectivity is still there. These are advantages over the probabilistic schemes. But deterministic schemes are less resilient to node capture; whole network is compromised after capturing a certain number of nodes (t) from the network. This is called t-collusion resistance. Polynomial-based key predistribution schemes [6, 7] also allow any pair of nodes in a network to be able to find a pairwise secret key. To improve the security of the polynomial based key distribution scheme, Liu and Ning [7] propose two schemes namely polynomial pool based key pre distribution and grid based key pre distribution.

In this paper, we propose a novel scheme for key distribution based on polynomials over finite fields. The proposed scheme has advantages of both probabilistic and deterministic approaches. As it is based on polynomials, for a node, it guarantees the establishment of keys with every neighbor. After initial deployment, polynomial from each node is erased. This makes our scheme very much resilient to node capture. Erasing polynomial gives rise to problem of adding new nodes in the network. To resolve this issue, we proposed a method for adding new node explained in Section II. The paper is organized as: In the next section, we discuss our scheme. Section III provides analysis of our algorithm. Section IV contains simulation studies and comparison with existing schemes. Finally, we conclude in Section V.

2. Our Approach

2.1. Architecture, assumptions and notations

We assume that the immediate neighboring nodes of any sensor node will not be known in advance. As wireless communication is not secure, we assume an adversary can eavesdrop on all the traffic, inject packets. The sensors nodes are randomly distributed and are not aware of the topology prior to the deployment. Sensor nodes remain stationary during the operation of the network. Being wireless and working unattended, often in a hostile environment renders sensor networks vulnerable to a number of attacks. Classical network attacks include eavesdropping, message interception, message reproduction and jamming. When nodes are captured, their memory can be read and erased or tampered with. Therefore, an adversary would know all the contents of a compromised node's memory. However adversary may eavesdrop on the traffic. A general assumption [9] is that an adversary does not capture a sensor node for a certain period of time following the network's initial deployment. So the network initialization takes place safely. There exists a lower bound on the time interval T_{min} that is necessary for an adversary to compromise a sensor node, and time T_{est} for a newly deployed sensor node to discover its immediate neighbors is smaller than T_{min}. In practice, we expect T_{est} to be of the order of several seconds, so it is a reasonable assumption that $T_{min} > T_{est}$.

Notations: We use the following notation to describe our key distribution scheme and cryptographic operations in this report:

– *U* and *V* are the communicating nodes.
– ID_U is the ID of a node *U*
– $K_{U,V}$ is the secret key between nodes *U* and *V* used for communication between these two nodes.
– *N* is the total number of nodes deployed initially
– *m* is the average number of neighbors that a typical sensor node has.
– *R* is the transmission range of a node
– $ENC_{K_{U,V}}(M))$ is the message *M* encrypted with key shared between the nodes *U* and *V*.

2.2. Algorithm for Initial deployment

Initially every node is given a symmetric polynomial P(X,Y) calculated on a value specific to each node. When nodes are deployed in the field then every node will find neighbors and establish a key with each of them by exchanging its ID with their IDs and calculating a common key between them. Node ID can be assigned by some central server before deployment. After this when we are adding new nodes then algorithm explained in the next section will be used.

Initial Setup: Before deployment a symmetric polynomial is calculated off-line *P(X, Y)*. Each node U is given this polynomial *P(X, Y)* evaluated on its node ID, that is, ID_U and this would be $P(ID_U, Y)$. After deployment each sensor node would find its neighbors and gets their IDs. If neighbors of a node *U* are $V_1, V_2, V_3, V_4, \ldots, V_n$ then calculated common keys will be like this:

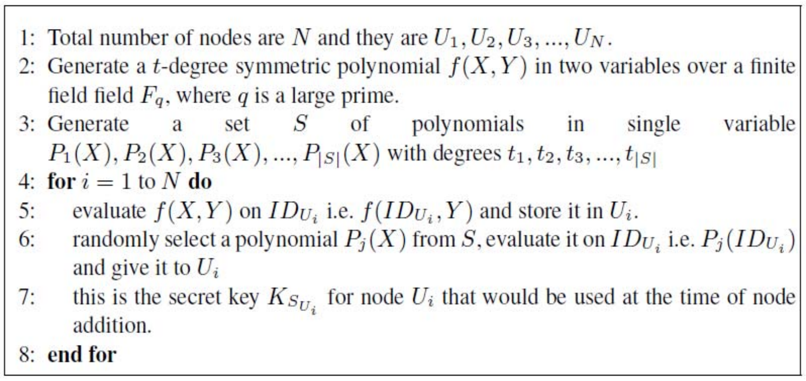
$K_{U,V1} = P(ID_U, ID_{V1}) = K_{V1,U}$

$K_{U,V2} = P(ID_U, ID_{V2}) = K_{V2,U}$

...

$K_{U,Vn} = P(ID_U, ID_{Vn}) = K_{Vn,U}$

Here, $K_{Vn,U}$ is the value evaluated by node V_n. As soon as keys have been calculated each node will erase its polynomial and every node would have only keys and no other information. In case there is a need to establish a key between two nodes that don't share a key, path key will be established using intermediate nodes. Fig. 1 and Fig. 2 present the algorithms describing how initial setup will take place and how pairwise keys will be established with the neighbors of a node respectively.



```
1: Total number of nodes are $N$ and they are $U_1, U_2, U_3, ..., U_N$.
2: Generate a $t$-degree symmetric polynomial $f(X,Y)$ in two variables over a finite
   field field $F_q$, where $q$ is a large prime.
3: Generate a set $S$ of polynomials in single variable
   $P_1(X), P_2(X), P_3(X), ..., P_{|S|}(X)$ with degrees $t_1, t_2, t_3, ..., t_{|S|}$
4: for $i = 1$ to $N$ do
5:    evaluate $f(X,Y)$ on $ID_{U_i}$ i.e. $f(ID_{U_i}, Y)$ and store it in $U_i$.
6:    randomly select a polynomial $P_j(X)$ from $S$, evaluate it on $ID_{U_i}$ i.e. $P_j(ID_{U_i})$
      and give it to $U_i$
7:    this is the secret key $K_{S_{U_i}}$ for node $U_i$ that would be used at the time of node
      addition.
8: end for
```

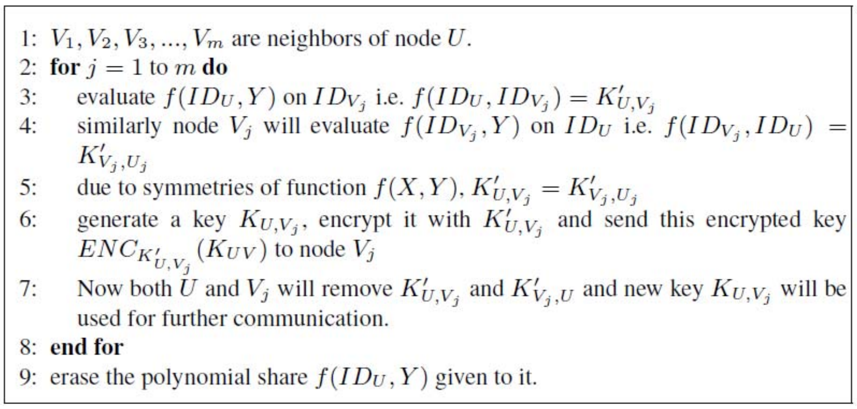
Fig. 1: Initial Setup for Network

Establishing Path Key: When a node U send data to a node V that is not a neighbor, a key $K_{U,V}$ has to be established between them. The node U would choose a key and send to its neighbors which would again forward it to next neighbor and finally reaching to the node V. Every node on the path has a key shared with its neighbors so nobody can get the key K_{UV} because it is encrypted with pairwise key between every link. Thus if path from U to V is $V_1, V_2, V_3, \ldots, V_c$ then key K_{UV} will reach like this:

$ENC_{K_{U,V1}}(K_{UV}) \rightarrow ENC_{K_{V1,V2}}(K_{UV}) \rightarrow ENC_{K_{V2,V3}}(K_{UV}) \rightarrow \ldots \rightarrow ENC_{K_{Vc-1,Vc}}(K_{UV}) \rightarrow ENC_{K_{Vc,V}}(K_{UV})$

One security threat is due to display of key K_{UV} at every intermediate node. This can be handled by employing existing multipath path key establishment scheme. U will find k disjoint paths to V. Then U will send the key K_{UV} to V through these disjoint paths breaking the key K_{UV} into k pieces as

$K_{UV} = r_1$ **XOR** r_2 **XOR** … **XOR** r_k. Each r_i will go from different path and whole key is not displayed to any node at intermediate level. At V key K_{UV} will be recalculated by XORing all the pieces like r1 **XOR** r2 **XOR**…**XOR** rk. This is done to improve the security of probabilistic key scheme. Now it is more secure with slight increase in communication overhead.



```
1: $V_1, V_2, V_3, ..., V_m$ are neighbors of node $U$.
2: for $j = 1$ to $m$ do
3:    evaluate $f(ID_U, Y)$ on $ID_{V_j}$ i.e. $f(ID_U, ID_{V_j}) = K'_{U,V_j}$
4:    similarly node $V_j$ will evaluate $f(ID_{V_j}, Y)$ on $ID_U$ i.e. $f(ID_{V_j}, ID_U) =$
      $K'_{V_j,U_j}$
5:    due to symmetries of function $f(X,Y)$, $K'_{U,V_j} = K'_{V_j,U_j}$
6:    generate a key $K_{U,V_j}$, encrypt it with $K'_{U,V_j}$ and send this encrypted key
      $ENC_{K'_{U,V_j}}(K_{UV})$ to node $V_j$
7:    Now both $U$ and $V_j$ will remove $K'_{U,V_j}$ and $K'_{V_j,U}$ and new key $K_{U,V_j}$ will be
      used for further communication.
8: end for
9: erase the polynomial share $f(ID_U, Y)$ given to it.
```

Fig. 2: Network Discovery for node U

2.3. Adding new nodes

No information is stored in the nodes after initial deployment. This poses a difficulty of adding new nodes. So, for addition of new nodes, we devise a novel way of adding nodes to the network. We are taking a set S of polynomials: $P_1(X)$, $P_2(X)$, $P_3(X)$, ..., $P_k(X)$ with degrees t_1, t_2, t_3, ..., t_k respectively. Before deployment, a polynomial from the pool S is chosen randomly. It is evaluated on the node ID and calculated value is given to the node. Algorithm for node addition is given in the figure.

```
1:  give all the polynomials $P_1(X), P_2(X), P_3(X), ..., P_{|S|}(X)$ to node $U$
2:  randomly select a polynomial $P_k(X)$ from $S$, evaluate it on $ID_U$ i.e. $P_k(ID_U)$
    and give it to $U$. This is secret key for node $U$ that will be used in future when
    some other node will be added as $U$'s neighbors.
3:  $V_1, V_2, V_3, ..., V_m$ are neighbors of node $U$.
4:  for $j = 1$ to $m$ do
5:      get $\alpha$ , $ENC_{K_{S_{V_j}}}(\alpha)$ and $ID_{V_j}$ for neighbor $V_j$ of $U$.
6:      for $i = 1$ to $|S|$ do
7:          evaluate $K'_{U,V_j} = P_i(ID_{V_j})$ that equals to $K_{S_{V_j}}$
8:          decrypt $ENC_{K_{S_{V_j}}}(\alpha)$ with $K'_{U,V_j}$ get decrypted value $\alpha'$
9:          if $\alpha' = \alpha$ then
10:             generate a key $K_{U,V_j}$, encrypt it with $K'_{U,V_j}$ and send as
                $ENC_{K'_{U,V_j}}(K_{U,V_j})$
11:             node $V_j$ will decrypt it with $K_{S_{V_j}}$ and will get the key $K_{U,V_j}$.
12:             erase $K = P_i(ID_{V_j})$
13:             break;
14:         end if
15:         erase $K = P_i(ID_{V_j})$
16:     end for
17: end for
18: erase all the polynomials
```

Fig. 3: Addition of a new node U

When a new node *U* is added, it is given the whole set *S* of polynomials. For key establishment, every neighbor V_j sends a value α and its encrypted value $ENC_{Pi}(ID_{Vj})(\alpha)$ encrypted with its secret key $Pi(ID_{Vj}) = K_{SVj}$, where $P_i(X)$ is the polynomial given to V_j initially. Node *U* will evaluate all the polynomials on the id ID_{Vj} of each neighbors V_j. U tries to decrypt $ENC_{Pi}(ID_{Vj})(\alpha)$ with all the values or keys calculated. It finds one such key $K'_{U,Vj}$ which gives α on decryption of $ENC_{Pi}(ID_{Vj})(\alpha)$. This is the common key between *U* and V_j by which new key would be sent. *U* randomly chooses a key $K_{U,Vj}$, encrypts it with $K'_{U,Vj}$ and sends encrypted value $ENC_{K'U,Vj}(K_{U,Vj})$ to V_j. U erases $K'_{U,Vj}$ once key $K_{U,Vj}$ is established between *U* and V_j . Node V_j decrypts $ENC_{K'U,Vj}(K_{U,Vj})$ and gets $K_{U,Vj}$. Now both *U* and V_j have the same key $K_{U,Vj}$ for further communication. Same process is repeated for each neighbors. After establishing keys with every neighbor, erase each polynomial.

3. Analysis of the Algorithm

We show below the complexity of evaluating a polynomial if nodes are captured.

Lemma: Let degree of each polynomial in the set S is t. Complexity of calculating a polynomial from the set S is O($^{(W+1)}C_{t+1} \times (t + 1)^3$) where W is the sum of the degrees of all the polynomial in the set S .

Proof: W will be equal to t × |S|. In order to find the polynomial of degree t an adversary has to capture at least W + 1 nodes. Because the adversary doesn't have any information about the secret key, so she has to apply brute force method only and consider all the subsets of size t+1 from all the W +1 nodes. There are $^{(W+1)}C_{t+1}$ such combinations possible. For each combination she can find a polynomial that fits to these values. The complexity of the procedure will be O($(t + 1)^3$). After finding a particular polynomial then the adversay needs to verify with other nodes to check whether the resultant is the original polynomial or not. Thus complexity of finding a particular polynomial is equivalent to solving a set of linear equations $^{(W+1)}C_{t+1}$ times. Thus the computational overhead for calculating a polynomial for a given set of values is shown in figure 4 and 5.

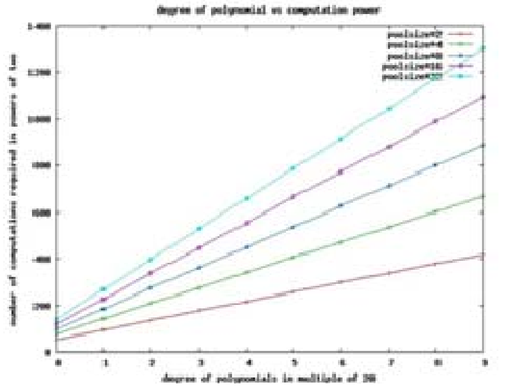


Fig. 4: Example

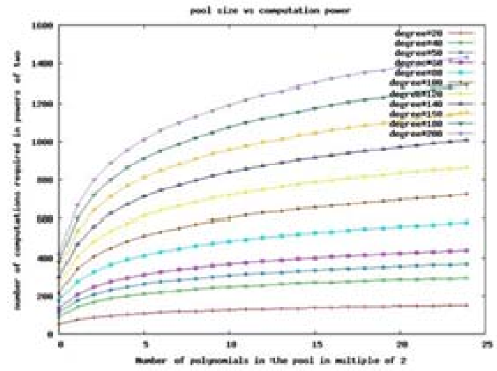


Fig. 5: Example

4. Simulation Studies and Comparison with Existing Schemes

For simulation, we take an area of $n \times n$ square meters and N number of nodes is scattered uniformly in the area. Thus coordinates (x, y) will be in the range from (0,0) to (n,n). Each node has an average m number of neighbors which is calculated as: $m = (n \times n)/(\pi \times R \times R)$ where each node has transmission range of R meters. Each node will send the detected event to some remote location. So we need to deploy $n-1$ special nodes on the line $y = n$, starting from $(R/2, n)$ to $((n-R)/2, n)$. We call these nodes as border nodes. These special nodes have more computational capacity and have more energy. Every node will send the messages to one of the border nodes to which message has to be sent and is decided on the least cost basis. Whichever node is closer to the sender, based on distance metric, is selected as the receiver and the message is sent to that node. We use Dijkstra algorithm for finding shortest path from sender to receiver.

4.1. Comparison with deterministic scheme

In the polynomial based key distribution scheme, it is easy to add a new node because every node has a polynomial share and that will be used to calculate the common key between two nodes. In our approach we solve the issue by giving every node U, a unique secret key K_{Su} that is used during addition of a new node. Although polynomial pool based key distribution scheme is better than the basic polynomial based scheme but it has one major drawback of increased memory overhead. If every node is given a set S of polynomials then space used to store these polynomials will be $s \times (t+1) \times logq$ where t is the degree of every polynomial. In our scheme, the space overhead is very less. Another advantage of our scheme has is that, it is very difficult to calculate a polynomial from the polynomial set $P_1(X), P_2(X), \ldots, P_{|S|}(X)$. While in polynomial pool based scheme, if a sufficient number of nodes are captured it is very easy to calculate the polynomials. The third advantage of our scheme is the network survival. Even if adversary captures any number of nodes, our network will survive and any existing link will not be compromised. The fourth advantage is that in pool based scheme when number of compromised nodes increases then fraction of compromised links between non-compromised sensors also increases but in our scheme no affect is observed on the links between non-compromised nodes. In our algorithm there will be some computational overhead initially. If there are m neighbors then in polynomial pool based and basic polynomial scheme, computation overhead required to calculate the polynomial m times but in our algorithm this will be $(|S|/2) \times m$. So we cannot keep the pool size $|S|$ very large. Typical values for $|S|$ and t are 8 and 50. For setting these values one requires 2^{230} computations to calculate a polynomial. Polynomial pool based scheme will not have any such calculation overhead once sufficient number of polynomials is captured. 8 polynomials of degree 50 will require a space of 8×50×64 bits *i.e.* 3200 bytes. However it will require 4 times computation overhead during initialization than previous schemes but it is comparable with the whole energy of a sensor node that will last for many days or months.

4.2. Comparison with probabilistic key distribution schemes

In the probabilistic key distribution, there is a trade-off between how many keys to be given to each node and what should be the size of the keypool. If number of keys given to a node is large then graph will be disconnected after a few node capture. If it is small then it will affect the connectivity of the network. In [1] it is shown that for a network of size 10,000 nodes for establishing a key with a node with probability 0.5 it is required to have 75 keys in each node. Here on an average every node will have 40 neighbors. If key size is 64-bit then space overhead will be 600 bytes and still we are getting connectivity with probability 0.5. But actually it needs to store keys with neighbors that will require 320 bytes space only. This is the space that is exactly required in our scheme. Although our algorithm will not have too much energy advantage over the probabilistic key schemes, still in the initial run our scheme will have a good percentage of energy advantage. Energy comparison of our scheme and probabilistic key scheme is plotted in figures 6-9 for different settings of number of nodes N , number of neighbors m of each node, area of the network in which nodes are deployed. Each graph is plotted for the ratio of energy consumed in the random scheme to energy consumed in our scheme. It is clear that there is some energy advantage in our scheme shown in the figure. Energy is less consumed initially in our scheme due to taking a direct path from source to destination. But in probabilistic key scheme path taken may not be the shortest path taken due key sharing restriction. Third advantage is, if nodes are captured in a sufficient number then whole network gets disconnected in probabilistic schemes. In our algorithm network will not get disconnected even after capturing any number of nodes as long as there exists nodes in the path. Key compromises will not let our network disconnected that is the case in probabilistic key schemes. Number of messages sent, to establish a path, is less in our scheme.

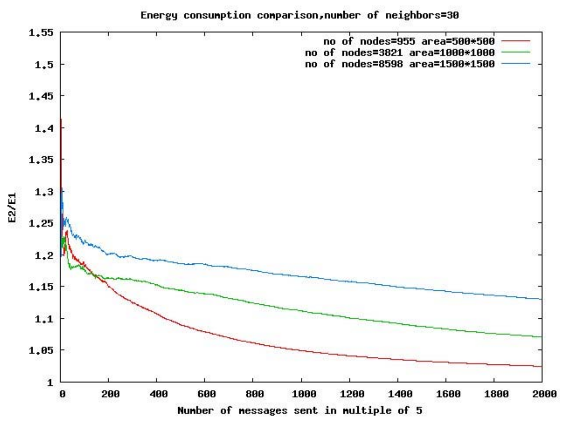


Fig.6: neighbor=30

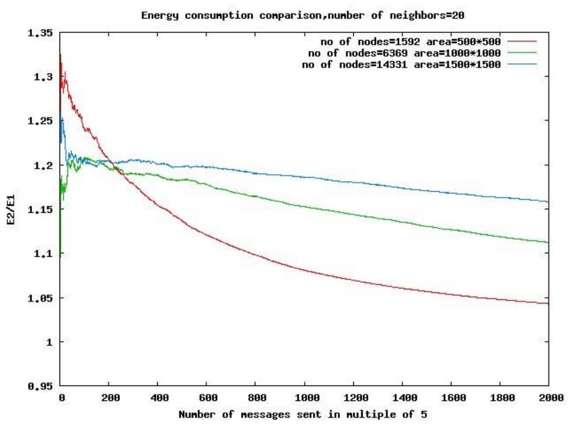


Fig.7: neighbor=50

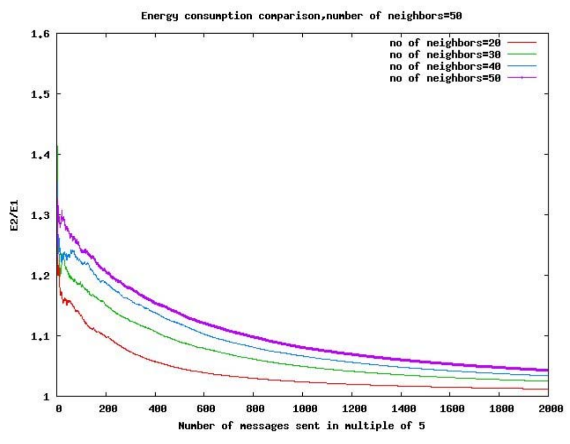


Fig.8: area=500x500

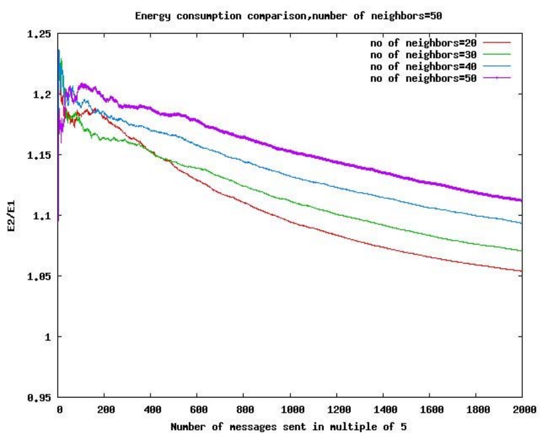


Fig.9: neighbor=1000x1000

5. Conclusion and Future work

We proposed a key distribution scheme that has a very good resilience to node capture. Any number of node's capture doesn't affect even a single link in the remaining network. Also it has many other advantages over probabilistic key distribution scheme and deterministic key distribution scheme. Compared to deterministic scheme, it is more secure, requires less space for keying material, have full network connectivity even after compromising a great fraction of nodes in the network and doesn’t have compromised links between non-compromised nodes. With respect to probabilistic key distribution schemes, our scheme provides full connectivity, consumes less energy and provides full network connectivity even after compromising any number of nodes from the network, requires less space and less number of messages to establish a key. Finally, very good resilience to node capture is the main advantage our scheme achieves. There is some scope to improve upon our algorithm. Developing a scheme, that provides better path key establishment while retaining good features of our algorithm, would be future work.

6. References

[1] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In: *9th ACM conference on Computer and Communications Security*, 2002.

[2] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In: *IEEE Symposium on Research in Security and Privacy*, 2003.

[3] R. Blom, An optimal class of symmetric key generation systems. *Eurocrypt,* 1976.

[4] Wenliang Du and Jing Deng and Yunghsiang S. Han and Pramod K. Varshney and Jonathan Katz and Aram Khalili, A pairwise key predistribution scheme for wireless sensor networks, *ACM Trans. Inf. Syst. Secur.*, 8, 2, 2003, 228–258, ACM, New York, NY, USA.

[5] J. Lee and D. Stinson. Deterministic key pre-distribution schemes for distributed sensor networks, *http://www.cacr.math.uwaterloo. ca/dstinson/ pubs.html*, 2004.

[6] C. Blundo, A. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. *Crypto*, 1992.

[7] D. Liu and P. Ning. Establishing Pairwise Keys in Distributed Sensor Networks. In: *10th ACM Conference on Computer and Communications Security (CCS ’03),* 2003.

[8] D. Liu and P. Ning, Location-based pairwise key establishment for static sensor networks, In: *1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003.

[9] Adrian Perrig and Robert Szewczyk and Victor Wen and David E. Culler and J. D. Tygar, SPINS: security protocols for sensor netowrks. *Mobile Computing and Networking*, 189–199, 2001