

Efficient Key Management in Sensor Networks

Abhishek Parakh and Subhash Kak
 Computer Science Department
 Oklahoma State University
 Stillwater, OK 74078
 Email: {parakh, subhash.kak}@okstate.edu

Abstract—We present an efficient key distribution scheme for sensor networks in which resource constraints and the possibility of node capture and malfunctions are considered. We deal with the problem of efficient routing and key distribution simultaneously. The resulting scheme is scalable and resilient to node captures and malfunctions. Under certain assumptions, the scheme has an upper bound of $\log_2 N$ on the diameter of the network and the number of keys required per node (where N is the number of nodes in the network).

I. INTRODUCTION

Wireless sensor networks are expected to be ubiquitous in a wide range of applications such as data gathering, patient monitoring, military applications and environment monitoring. When the data collected is of sensitive nature, the nodes need to communicate with other nodes to send the data to the sink or perform distributed computation and this requires communication to be protected using cryptographic methods. The use of public key cryptosystems for sensor networks is ruled out because of large energy consumption [11], and one secret master key for all sensor nodes, although infinitely scalable, is insecure for obvious reasons. An all-pairs unique key distribution, requiring every sensor to store $N - 1$ keys is infeasible because of the memory constraints on sensor nodes [3] and the difficulty and inefficiency of adding new nodes into the network. Hence symmetric key pre-distribution is the usual approach.

It is common to model sensor networks as random graphs based on the assumption that there is no a priori deployment knowledge and the sensors have a very limited communication range [5]. Although, this model is general and widely applicable, it is not representative of many situations. This model involves randomly distributing keys to sensors from a large pool of keys S such that any two neighbors share at least one key with probability p and the whole network (graph) is connected with a certain probability. To achieve this, each node carries a subset of keys, called a key ring, of size m from S and the problem is then to find a suitable intersection between any two randomly chosen subsets. Such a scheme requires three steps, namely, key pre-distribution, key discovery, and path key establishment. The key pre-distribution is explained above. In the key discovery phase, every node broadcasts its key IDs to all its neighbors and upon receiving the broadcast from others, determines which keys are shared with which node. In the path key establishment phase, two nodes that do not share a key establish a key via multihop.

A q -composite scheme was proposed by Chan *et al.* [2] in which they increased the requirement of number of keys that are shared by neighbors to $q > 1$ resulting in increased network resilience against node captures but requiring a decrease in the key pool size $|S|$, in turn increasing the probability of key repetition or an increase in the key ring size at each node. Further, the q composite scheme is only advantageous when the number of nodes captured is below about $0.01N$, where N is the total number of nodes in the network, beyond which it becomes disadvantageous. Other schemes assume deployment knowledge [13], [3]; Du *et al.* [3] propose a scheme assuming deployment knowledge modeled based on Gaussian distribution, which is an arbitrarily chosen probability density function. Other pdfs may be chosen resulting in different analysis. Also Du's scheme only allows for static nodes because of their assumed probability density function for the deployment of sensors.

A pairwise key distribution scheme is developed by Du *et al.* in [4], where they build upon the idea of Blom's key distribution scheme that is λ resilient, which means that if less than $\lambda + 1$ nodes are compromised, the uncompromised nodes remain secure and if more than λ nodes get compromised, the entire network is compromised. Such a scheme requires $O(\lambda)$ memory spaces to store the generator matrix and a secret set of numbers that will be used to generate keys with neighboring nodes. A large λ increases network resilience, it also increases the memory requirement and vice versa. The scheme has a local computational overhead and the degradation of network characteristics is not gradual which may not be acceptable in many applications. Further, if an adversary captures a node and masquerades as that node, then it can potentially perform a key exchange with every other node in the network.

A scheme by Jarraj and Saifan [1] adopts a hybrid approach by combining random key pre-distribution with local key exchange via neighbors. Such a scheme minimizes the interaction between nodes and the base station. Other schemes for key pre-distribution and discussion on framework for it can be found in [9], [8], [7].

The scheme that comes closest to our scheme is the random pair wise key distribution scheme by Chan *et al.* [2]. In Chan's scheme, pairs of nodes are picked at random from the network and unique keys are assigned to the pairs. This provides mutual node authentication because of uniqueness of the key shared and it eliminates the key discovery phase. Like most key distributions schemes, it focuses on network connectivity while

finishing the routing problem.

In this paper, we confront the joint problem of key distribution and secure message routing. We work under the assumption that almost all the sensor nodes are within communication range of each other. At first blush, it may seem that our assumption precludes the need for a routing algorithm. But note that we are only interested in secure transmission of messages, and although all the nodes can communicate with each other in plaintext, secure communication can happen only between nodes that share keys. If we use a random key distribution scheme for a network that follows our assumption, two possible methods exist for secure message transmission. First, the sensors could build the adjacency matrix of the network based on shared keys. In order to do this, every sensor would have to communicate to other sensors with whom they share keys and then run a shortest path algorithm. Second, sensors could encrypt every message they receive that is not intended for it with all the keys that they have in their key ring and flood the network. However, this is not only computationally expensive, it also requires a time-to-live for each message and keeping track of already visited nodes in order to avoid loops. Hence, a random key distribution scheme is unable to take advantage of the fact that all the sensors are within communication range. We present a strategy that takes advantage of our assumption and presents a balanced solution for routing and key distribution.

When compared to Chan's work [2], we pick node pairs from the network and assign them unique keys, but we pick them based on certain logic that makes the routing of messages efficient.

Our contributions are:

- 1) The use of an overlay architecture for key distribution that makes the secure routing of messages between sensors efficient.
- 2) An efficient scheme that requires $O(\log_2 N)$ keys per node with the upper bound on the maximum path length being $O(\log_2 N)$.
- 3) Resilience against node captures: one node capture only affects that node and its connections, no other node communication is compromised.
- 4) Scalability and resilience against link and node failures.

II. THE PROPOSED KEY DISTRIBUTION SCHEME

Assume that the nodes are assigned node IDs in a sequence from 1 to N . The nodes in the network can then be (logically) assumed to lie on a circle according to their node IDs as shown in figure 1 for a network of size 16. To simplify our explanation we will assume our networks to be of size 2^k for some integer k .

The key pre-distribution is performed as described: node with node ID i shares unique keys with nodes $i+1$, $i+2$, $i+4$, $i+8$, and so on until $i + \frac{N}{2}$, computed modulo N .

Every shared key is represented by an edge between nodes in figure 2 and can be used for secure communication. Since the keys distributed are symmetric keys, the same key may be used for communication in both directions and the nodes in the

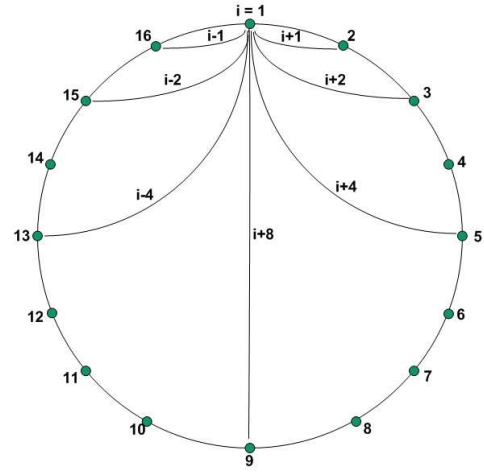


Fig. 1. Only connections of node 1 are shown in a network size of $N = 16$.

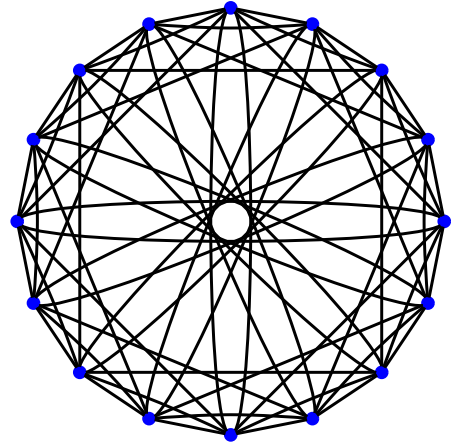


Fig. 2. All the connections shown in a network size of $N = 16$.

resulting network can securely communicate with $2\log_2 N - 1$ nodes directly, while the network has a total of $N\log_2 N - \frac{N}{2}$ keys.

The messages can be securely routed using direct connections or via multi-hop using intermediate nodes. The routing can follow three procedures,

- 1) Treating the connections in the graph as bidirectional
- 2) Treating the graph to be made of two halves of $\frac{N}{2}$ nodes symmetrically arranged on the two sides of node i and assuming that the connections are unidirectional (clockwise in one half and anticlockwise in the other)
- 3) Assuming that the connections are unidirectional (clockwise)

Let N_i denote the set of direct neighbors of node i . Direct neighbors are those nodes which share a key. If a source node s wishes to send a message to a destination node d securely, the routing of messages takes place using one of the following algorithms. Let $dist(i, j)$ return the distance between nodes i and j by computing $\min((i - j) \bmod N, (j - i) \bmod N)$.

Algorithm 1 (Bidirectional Connections)

- 1) If $d \in N_s$, node s sends the message directly to d using the key they share
- 2) If $d \notin N_s$, node s sends the message to a node $m \in N_s$ encrypting the message using the key it shares with m , where m is chosen such that $dist(m, d)$ is minimum $\forall m \in N_s$
- 3) Upon receiving the message destined for d , node m decrypts the message and repeats step 1 and 2

Algorithm 2 (Unidirectional - 2 halves)

- 1) If $d \in N_s$, node s sends the message directly to d using the key they share
- 2) If $d \notin N_s$ and $s < d \leq s + \frac{N}{2} \text{ mod } N$ on the circle. Node s sends the message to a node $m \in N_s$ encrypting the message using the key it shares with m , where m is chosen such that $dist(m, d)$ is minimum $\forall m \in N_s$ and m appears before (clockwise) d on the circle
- 3) If $d \notin N_s$ and $s + \frac{N}{2} < d \leq (s-1) \text{ mod } N$ on the circle. Node s sends the message to a node $m \in N_s$ encrypting the message using the key it shares with m , where m is chosen such that $dist(m, d)$ is minimum $\forall m \in N_s$ and m appears before (anticlockwise) d on the circle
- 4) Upon receiving the message destined for d , node m decrypts the message and repeats step 1, 2 and 3

Algorithm 3 (Unidirectional Connections)

- 1) If $d \in N_s$, node s sends the message directly to d using the key they share
- 2) If $d \notin N_s$, node s sends the message to a node $m \in N_s$ encrypting the message using the key it shares with m , where m is chosen such that $dist(m, d)$ is minimum $\forall m \in N_s$ and m appears before d in the clockwise direction
- 3) Upon receiving the message destined for d , node m decrypts the message and repeats step 1 and 2

The maximum path length for algorithm 1 is $\lfloor \frac{\log_2 N}{2} \rfloor + 2$. This is because the connections *slice* the network into pieces doubling in sizes. The arrangement of connections therefore results in the largest slice of size $\frac{N}{4}$. In the worst case, the source node sends the message to a node that is at the edge of the largest slice. After the first hop the distance from the destination reduces to $\frac{1}{2} \times \frac{N}{4} = \frac{N}{8}$. From second hop onwards, the network no longer forms a closed circle and the connections only divide the remaining part of the network into slices such that the largest slice is of size $\frac{1}{2} \times \text{remaining part of the network}$, which after the first hop is $\frac{1}{2} \times \frac{N}{8} = \frac{N}{16}$. The second hop only reduces the distance to $\frac{M}{4}$, where M is the size of the remaining network. Continuing in this way, the upper bound on the number of hops for a network of size N is $\lfloor \frac{\log_2(\frac{N}{8})}{2} \rfloor + 2$. A formal proof may be found in [10].

The maximum path length for algorithm 3 is $\lceil \log_2 N \rceil$. This is because if the connections are considered unidirectional clockwise, then the distance from source to destination reduces

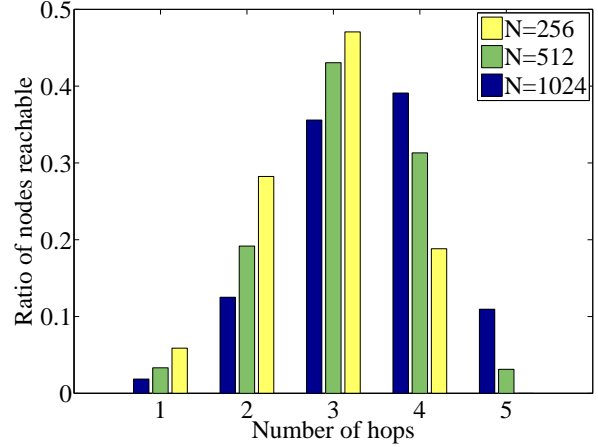


Fig. 3. Path length to neighbors

by powers of 2. This has been proven in [12] and is deducible from the procedure for binary search.

Algorithm 2, considers the network to have unidirectional connections, but be formed of two symmetric halves. The arguments for the maximum number of hops from the source to the destination that apply to algorithm 3, applies here as well but with half the size of the original network, making the upper bound to be $\log_2 N - 1$.

We consider Algorithm 1 as the primary algorithm for our network. Algorithm 2 and 3 may be used to find alternate paths in the case of link and node failures.

Another possible routing may be to send the message to every node within a pre-specified distance from the destination which are source node's direct neighbors.

Figure 3 shows the ratio of nodes reachable at various hops for different network sizes treating the links as bidirectional using Algorithm 1.

III. PRUNING THE NETWORK

We note that optimality is achieved when the number of keys per node is $\log_2 N$ and the maximum path between nodes is $\log_2 N$. Since every node has $2\log_2 N - 1$ connections resulting in a diameter much lower than $\log_2 N$. Consequently, we prune the network such that node i shares a key with nodes $i+1, i+4, i+16, i+64$ and so on, where alternate connections are removed.

Figure 4 shows the plot for average path length for the pruned network and the original network with bidirectional connections. We see that the deterioration in the path length is gradual and becomes prominent only for large network sizes. Other figures in the paper are for analysis on the un-pruned network and the simulations are run over 10 iterations. The simulations were performed in C and MATLAB and the sensor network was modeled using adjacency matrix.

IV. LINK AND NODE FAILURES

We assume that at any given time every node in the network is aware of which nodes have failed. Further, since entire

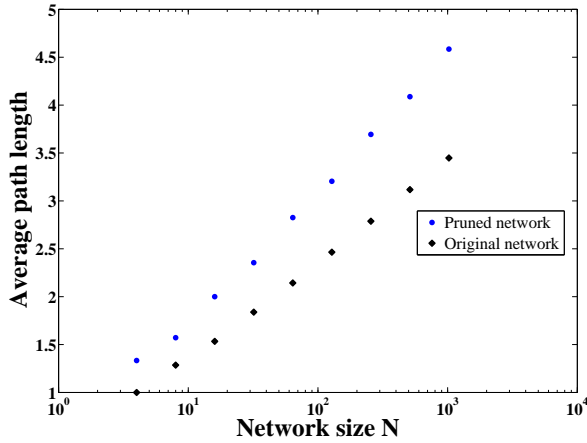


Fig. 4. Comparison of average path lengths for the original and the pruned network for varying network sizes.

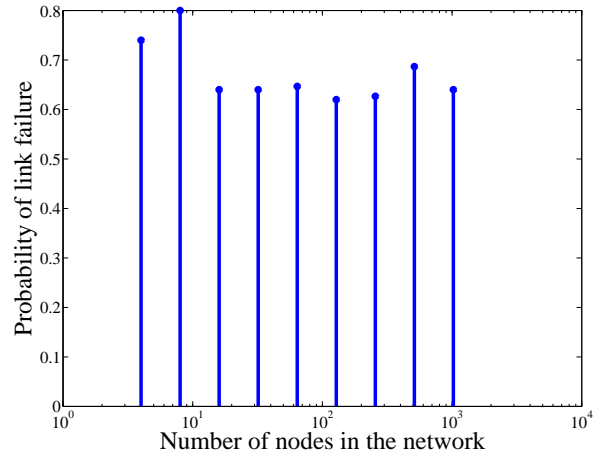


Fig. 6. Probability of link failures at which first isolated node appears for different network sizes.

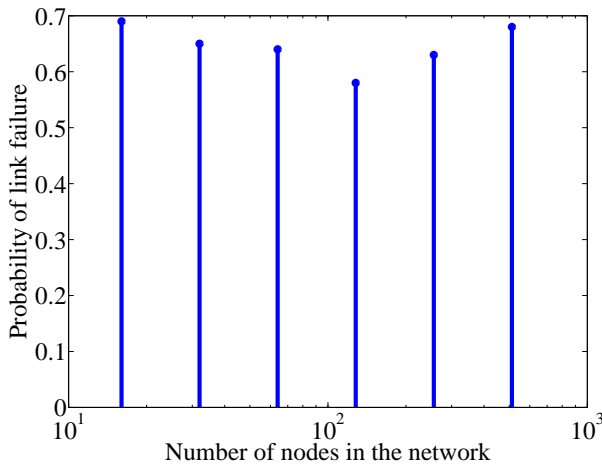


Fig. 5. Probability of link failures at which the network gets disconnected for different network sizes.

network's adjacency matrix is already known, a route can be efficiently computed from a source to destination. Once a node failure is detected, the node that detects this failure broadcasts a message to other nodes.

We performed link failure analysis and the results are plotted in figures 5 and 6. The simulations were performed by first establishing the connections as described before and then breaking these connections based on some probability of failure p_f . We observe the effect of link failures on connectivity of the network in figure 5 which shows that link failure probability may be as high as 0.55 before the connectivity of the network gets affected. When viewed along with figure 6, it shows that network connectivity and node isolation follow each other closely which implies that the network is resistant to breaking into smaller networks.

Despite assuming that nodes in the network are aware of which nodes have failed, some failures may occur while the message is in transit or the information of the node failure

has not yet reached the current source node. In order to cope with such situations, routing algorithms must be modified with an additional constraint that a destination upon receiving a message, sends back an acknowledgement to the source node. If the source node does not receive an acknowledgement in a pre-specified time interval, the source retransmits the message via an alternate path.

A. Routing Algorithm Failure

It is clear from figure 2 that there exist several alternate shortest paths and that Algorithm 1 determines some of them. If we consider the clockwise movement of messages as +ve hops and anti-clockwise movement as -ve hops, then the minimum number of hops required is the minimum number of powers of 2 needed to be added and subtracted so as to express the distance between the source and the destination.

For example, consider a network of size $N = 32$. Let node 1 be the source and node 13 the destination. Then Algorithm 1 dictates that it takes at least 3 hops to reach node 13 from node 1 and the intermediate nodes that the message may go through are 16 and 12. Hence, one of the paths computed by Algorithm 1 is $\langle 1, 16, 12, 13 \rangle$. The possible powers of 2 in this network are $2^0, 2^1, 2^2, 2^3, 2^4$ and Algorithm 1 expresses the distance between node 1 and node 13 as addition and subtraction of these powers, i.e. $13 = 16 - 4 + 1 = 2^4 - 2^2 + 2^0$. Further, the expression above dictates the distance between hops that the message needs to cover. Thus, +16 means the message should be first sent to 16^{th} node in the clockwise direction (reaching node 16) and -4 indicates that message should be next sent to 4^{th} node in the anticlockwise direction (reaching node 12), and the final +1 says that the message should be sent to the next node in clockwise direction (reaching node 13).

The path determined by Algorithm 1 is the shortest but not unique. Not all the nodes in the network need to compute the entire path, but only the next hop. Following the example above, the hops could be rearranged in $3!$ different ways,

namely $-4+16+1$, $-4+1+16$, $16+1-4$, $1-4+16$, $1+16-4$ and $16-4+1$. All of these paths are of same length and reach node 13. Further, the distance 13 cannot be expressed using fewer than 3 powers of 2 and hence requires 3 hops. A detailed analysis of the ‘binary search problem’ that is solved by Algorithm 1 can be found in [6] which also proves that Algorithm 1 finds a shortest path.

If paths are allowed to fail in our model then the probability that routing will fail is given by the probability of at least one intermediate link failing. This is expressed below,

$\Pr(\text{routing failure}) = 1 - \Pr(\text{no link fails on the path}) = 1 - (1 - p_f)^h$, where, h is the path length or number of hops and p is the probability of failure.

The paths may fail due to change in network topography, physical barriers or partial malfunction of nodes. Note that this is different from node failures, where one node failure affects at least $2\log_2 N$ links. A link failure is the failure of only a single link independent of other links, while the nodes at either end of the failed link may remain active.

In our example above, if probability of link failure is $p_f=0.05$, a path will fail with a probability of $1 - (1 - 0.05)^3 = 0.142625$, i.e. about 15% of the time.

Algorithm 1 also determines a number of alternate paths that may be taken, although not all of them are disjoint. For example, other paths to 13 may be $16-2-1$, i.e. $\langle 1, 16, 14, 13 \rangle$ and all permutations of $16, -2, -1$. We see that node 16 is common in some of the paths.

In general, longer paths have larger probability of failing than shorter ones. However, for longer paths many more alternate paths of comparable length exist. Since, the longest shortest path is of length $\log_2 N$, the probability of failure of this path is $1 - (1 - p)^{\log_2 N}$ and there exist at least $(\log_2 N)!$ alternate paths.

Other possible paths are given by Algorithm 2 and 3, which may be longer compared to Algorithm 1, but have an upper bound of $O(\log_2 N)$.

Another mechanism to handle routing failures is that - when a particular node detects that a link between itself and the next node on the path has failed, it finds an alternate path itself, instead of sending back a routing failure notification to the sender. Thus the current node now acts as the new source for the message. If necessary, when no next node is within reach according to the routing algorithm, the current node may step back one hop and compute all the nodes that are at a distance of one additional hop from the destination. This is possible because every node is aware of the network topology and it can then transmit the message to the next best node. The expansion of possible set of next hops only takes place until a predetermined radius beyond which the node sends a delivery failure notification to the source node.

Figure 7 analyzes the increase in the average path length between nodes until the first isolated node appears. We see that the increase is gradual and can tolerate high probability of link failure.

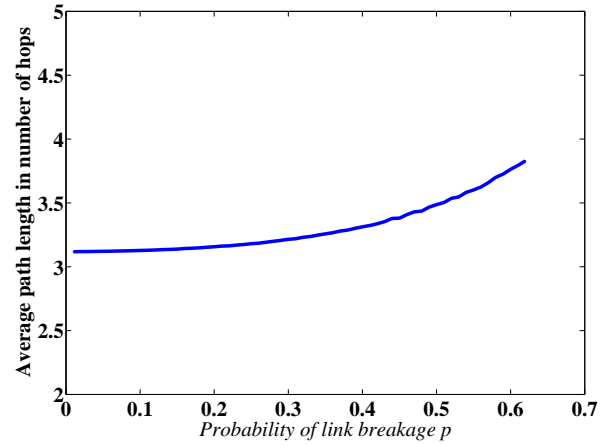


Fig. 7. Average path length between nodes and its degradation before the first isolated node appears ($N=512$).

V. ADDING NEW NODES

In order to enable the addition of new nodes to the sensor network, we adopt a strategy similar to that adopted in [12]. We assume that nodes are mapped into a larger ring with more number of node places than the actual number of nodes, such that some of these places remain unoccupied. These unoccupied spots should be uniformly distributed over the network. Another way to look at it is to assume that the ring initially had all points occupied, but eventually some nodes have failed leaving vacant spots. The original nodes in the network still share keys with nodes that previously occupied these vacant spots, which may now be taken up by new replacement nodes joining the network. The new nodes are preloaded with relevant shared keys. As soon as a new node joins the network, it broadcasts its presence to other nodes, enabling them to update their tables.

The number of vacant spots that may be left in the network can be determined by node failure analysis and its tradeoff against degradation of network connectivity. From the figure 8 we see that the network remains connected for probability of node failure as high as 0.43, indicating that as many as 40% of the nodes can fail without effecting network connectivity. This directly indicates the extent of scalability that can be achieved in the network. That is, by leaving as many as 40% of positions on the ring vacant, we can still guarantee network performance and add new nodes later to fill the unoccupied spots. For example, a network with 600 nodes may be mapped onto a circle of size 1000; consequently as many as 400 nodes may join the network.

A note on scalability: Since a node shares keys with $2\log_2 N - 1$ other nodes, only $O(\log_2 N)$ new keys have to be distributed when a new node joins the network.

VI. LIMITING THE COMMUNICATION RANGE OF NODES

Assume that the nodes are randomly deployed and that their communication range is limited to k neighbors, which is a fraction of the total nodes in the network. This is the scenario

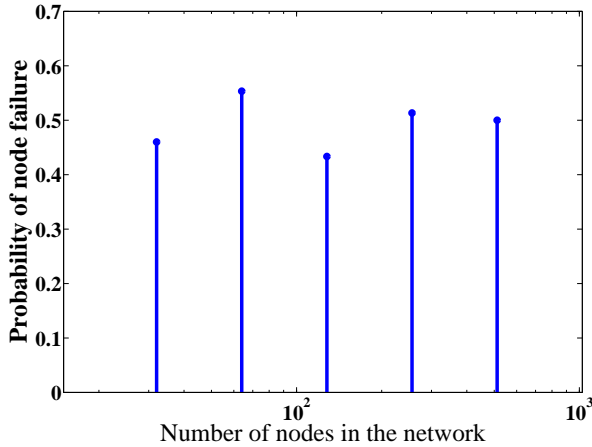


Fig. 8. Probability of node failures at which the network becomes unconnected against different network sizes.

assumed by most of the previously proposed schemes. Further, the minimum number of edges per node (degree d) required to guarantee network connectivity with a high probability can be determined by first deriving the minimum probability of connection, p , required between nodes (using the Erdos - Renyi formula [5]), and then computing $d = p(N - 1)$.

In our model every node has specific number of connections $m = 2\log_2 N - 1 > d$. The probability that a node i with k neighbors, $k > m$, will share keys with exactly d of them is computed by noting that there are C_k^m ways to choose k neighbors. There are C_d^m ways to select d neighbors such that they share a key with node i and there are C_{k-d}^{N-m} ways to select remaining neighbors with whom there is no shared key, where $C_b^a = \frac{a!}{(a-b)!b!}$. Therefore, the probability that out of k neighbors, exactly d of them share a key with node i is given by $p(d) = \frac{C_d^m \times C_{k-d}^{N-m}}{C_k^m}$. Given a fixed k , the larger the m , the better the probability of graph being connected. The probability that the graph will be connected is given by $\sum_j p(j)$, $d \leq j \leq m$.

If $d > m$, that is the required degree is larger than the number of connections, then new connections may be established by connecting node i to a node in the middle of the largest slice in the network. For example, a new connection for node i may be established between $i + \frac{N}{2}$ and $i + \frac{N}{4}$ the first time and for subsequent new connections one may divide the largest remaining slice of the network into two halves each time.

In sensor networks, local communication is more common than communication across the entire network, therefore, a random sensor network may be modeled as consisting of smaller well connected networks.

VII. CONCLUSIONS

We have presented an efficient key distribution method for sensor networks that has applications in wireless mesh networks and computer networks as well, and has an efficient routing algorithm with a bound of $O(\log_2 N)$ on the path

length. We have analyzed the effect of link failures and node failures on network connectivity. We have proposed methods for coping with such failures and for adding new nodes to the network. Lastly, the effects of limited communication range of nodes has been discussed.

Future work involves studying the effects of mobile sensor networks and extension of our scheme to a two dimensional layout, which is more intuitive. Also, in the case of random deployment, one needs to determine how subnetworks (of limited randomness) may be formed that have optimal characteristics within themselves and application of our scheme to it may be investigated.

REFERENCES

- [1] O. Al-Jarrah and R. Saifan. A novel key management algorithm in sensor networks. In *MoMM '08: Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia*, pages 291–294, New York, NY, USA, 2008. ACM.
- [2] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, page 197, Washington, DC, USA, 2003. IEEE Computer Society.
- [3] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. volume 1, page 597, march 2004.
- [4] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili. A pairwise key predistribution scheme for wireless sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(2):228–258, 2005.
- [5] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47, New York, NY, USA, 2002. ACM.
- [6] P. Ganesan and G. S. Manku. Optimal routing in chord. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 176–185, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.
- [7] J. Hwang and Y. Kim. Revisiting random key pre-distribution schemes for wireless sensor networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 43–52, New York, NY, USA, 2004. ACM.
- [8] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 52–61, New York, NY, USA, 2003. ACM.
- [9] Z. Liu, J. Ma, Q. Huang, and S. Moon. A pairwise key establishment scheme for heterogeneous sensor networks. In *HeterSanet '08: Proceeding of the 1st ACM international workshop on Heterogeneous sensor and actor networks*, pages 53–60, New York, NY, USA, 2008. ACM.
- [10] A. Parakh and S. Kak. A key distribution scheme for sensor networks using structured graphs. In *ELECTRO-2009, Banaras Hindu University, Banaras, India, 2009*.
- [11] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. Spins: security protocols for sensor networks. *Wirel. Netw.*, 8(5):521–534, 2002.
- [12] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.
- [13] Z. Yu and Y. Guan. A key management scheme using deployment knowledge for wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst.*, 19(10):1411–1425, 2008.