

Source Routing Based Pairwise Key Establishment Protocol for Sensor Networks

Dijiang Huang Manish Mehta Deep Medhi

Computer Science and Electrical Engineering Department
University of Missouri–Kansas City
Kansas City, Missouri 64110, USA.
{dhuang, manish.mehta, dmedhi}@umkc.edu

Abstract

Sensor networks are composed of a large number of low power sensor devices. For secure communication among sensors, secret keys must be established between them. The establishment of secret keys after deployment of sensors requires wireless communication. Because of the energy constraints, an efficient key establishment scheme cannot be designed without considering the power consumption factor in wireless communication. In order to reduce the communication overhead, we propose a source routing based pairwise key establishment protocol for large-scale sensor networks. We then use a probability model proposed in [1] to analyze the communication overhead, and security strength of the scheme.

1 Introduction

Sensor networks are composed of a large number of low-power sensor devices. Typically, these networks are installed to collect sensed data from sensors deployed in a large area. Within a network, sensors communicate among themselves to exchange data and routing information. Because of “wireless” nature of communications among sensors, sensor networks are vulnerable to various active and passive attacks on communication protocols. This demands secure communication among sensors.

Typically, sensor networks are deployed with large quantities of sensor devices. According to [2], the number of sensor nodes deployed in studying a phenomenon may be in the order of hundreds or thousands; depending on the application, the number may reach millions. Due to inherent storage constraints, it is infeasible for a sensor device to store a shared key for every other sensor in the system. Also, because of the lack of post-deployment geographic configuration information of the sensors, keys cannot be selectively stored in sensor devices. Random Key Predistribution

(RKP) schemes ([3, 4, 5, 6, 7]) have been proposed to provide flexibility for the designers of sensor networks to tailor network design to the available storage and the security requirements. After deployment of sensors, each sensor discovers key sharing relations among all its neighbors. The shared key discovery protocols has been described in [3] and [4]. Since RKP schemes have limited number of keys pre-installed in sensors, a sensor may not share a key with all of its neighbors. In this case, a Pairwise Key Establishment (PKE) protocol is required to set up pairwise keys with all its neighbors. In all recent proposals, during the PKE phase, the intermediate nodes may not necessarily be located within the source node’s communication range. Thus, the source may not be able to determine paths to set up pairwise keys with its neighbors. In other words, the key setup requests must be flooded instead of using chosen paths. Consequently, the communication overhead invoked by the pairwise key requests can be prohibitively high. Studies in [1] show that the key path length (number of forwarding hops) for PKE is longer when the key graph (is defined in Section 2.1. connectivity is low. Thus, flooding the key setup requests results in high communication overhead.

Studies in [8] show that the energy consumption due to communication in sensors is several orders higher than that due to computation overhead. In order to reduce the communication overhead, we propose a source routing based PKE protocol in which the sensors set up pairwise keys using only their neighbor nodes. In other words, each sensor can choose paths to set up pairwise keys with its neighbors. This design significantly reduces the communication overhead involved in the PKE phase. Similar to the recent schemes in [3, 4, 5, 6, 7], our model is based on networks with uniformly distributed sensors. We give an analysis of communication overhead due to the proposed PKE

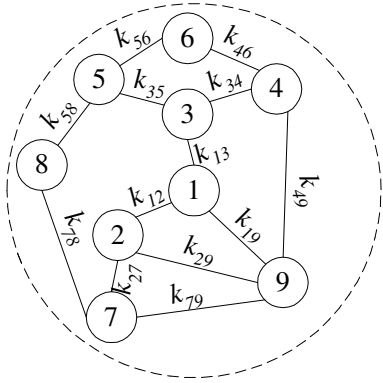


Figure 1: Key graph example (centered with node 1)

protocol.

The rest of the paper is organized as follows: An energy-efficient pairwise key establishment protocol is described in Section 2. A performance analysis addressing communication overhead, security strength, and energy consumption for the protocol is given in Section 3. Section 4 concludes the work and provides future research directions.

2 An Energy-efficient Pairwise Key Establishment Protocol

2.1 Key Graph

Definition 1 (Key Graph) A key graph maintained by node i is defined as $G_i = (V_i, E_i)$ where, $V_i = \{j | j \text{ is a neighbor of } i \text{ and } j = i\}$, $E_i = \{e_{jk} | j, k \in V_i \text{ and } k \text{ is a neighbor of } j, \text{ and } j \text{ is a neighbor of } k, \text{ and } j \text{ shares at least one key after the shared key discovery phase with } k\}$.

In Figure 1, we present an example of key graph, which is centered by node 1. The dash line represents the communication range of node 1 and the solid line represents the shared key relation between nodes.

2.2 Protocol Overview

In [4, 6, 3, 5, 7], the authors assume that there exists a key discovery protocol (hello protocol) that a sensor can rely on to discover all its neighbors and the shared key relations among its neighbors. After using the key discovery protocol, a node, say i , knows (1) all nodes located within its range (set W_i), we call them W-neighbors of i ; (2) all its W-neighbors who share at least one key with it (set Q_i), we call them Q-neighbors of i ; (3) all its W-neighbors who do not share key with it (set R_i), we call them R-neighbors of i ; and (4) a *key graph* derived from W_i and Q_i . We assume that all sensors are uniformly deployed in a given two-dimensional area. After the key discovery phase, each

Requests (Type 1):

$\langle 1 \rangle E_{k_{12}} \{7\}$

$\langle 2 \rangle E_{k_{13}} \{\{4,6\}, \{5,8\}\}$

Responses (Type 2):

$\langle 3 \rangle E_{k_{27}} \{k_{17}\}$

$\langle 4 \rangle E_{k_{12}} \{k_{17}\}$

$\langle 7 \rangle E_{k_{13}} \{k_{15}\}$

$\langle 8 \rangle E_{k_{13}} \{k_{14}\}$

$\langle 9 \rangle E_{k_{38}} \{k_{19}\}$

$\langle 10 \rangle E_{k_{38}} \{k_{16}\}$

$\langle 11 \rangle E_{k_{15}} \{k_{19}\}$

$\langle 12 \rangle E_{k_{15}} \{k_{16}\}$

Responses&Request (Type 3):

$\langle 5 \rangle E_{k_{34}} \{\{k_{15}\}, \{8\}\}$

$\langle 6 \rangle E_{k_{34}} \{\{k_{14}\}, \{6\}\}$

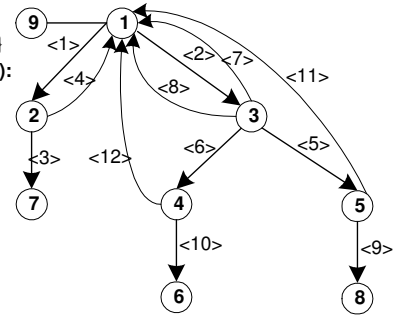


Figure 2: Illustration of pairwise key establishment protocol, where E is encryption algorithm and k_{xy} represents the pairwise key shared by nodes x and y .

sensor has n' W-neighbors and $(n' \cdot p_1)$ Q-neighbors, where p_1 is the probability that two sensors share at least one key [3]. Initially, $f = \frac{|Q_i|}{|W_i|} = p_1$. The goal of our proposed PKE protocol is to achieve $f > c$, where f is greater than a threshold value. This means that, on an average, a sensor will set up pairwise key with $c\%$ percent of all its W-neighbors, where the value of c depends on type of applications.

Due to physical limitations of sensors, we assume that the sensors are stateless for all the communication related to pairwise key setup. A sensor who helps other sensors to set up pairwise key will not maintain the state of the connections. Only the source of the request message maintains a timer t and retry threshold T for a subset or all of its R-neighbors. During the PKE procedure, a sensor sends out the pairwise key setup requests and waits for responses. If no response is received for a request before timer t expires, the sensor will resend the request until it reaches the retry threshold T .

2.3 Pairwise Key Setup Request And Response

During the PKE phase, a sensor sends key setup requests asking its Q-neighbors to help establish pairwise keys with its R-neighbors.

We illustrate the process of our proposed PKE protocol using the example scenario shown in Figure 1 and we draw the shortest path tree in Figure 2. In this example, node 1 has 8 neighbors. The links and nodes represent the shortest path tree that node 1 derives from its *key graph*. The link directions represent the propagation direction of pairwise key requests and responses (node 9 does not have offsprings in the tree;

thus, it cannot help node 1 to set up pairwise key with other nodes). The encrypted messages for all transmissions are shown on the left side in Figure 2. In the example, node 3 receives the request from node 1; the request $\{\{4, 6\}, \{5, 8\}\}$ is encrypted under the pairwise key k_{13} used by the pair (1, 3); node 1 requests for setting up pairwise keys with node 4, 5, 6, and 8. Since every node broadcasts its Q-neighbor list during the key discovery phase, node 1 knows the exact paths to reach its R-neighbors. Since other nodes, for example, node 3, may not maintain the same shortest path tree as that of node 1, node 1 needs to inform the nodes in the *key paths*¹ in which directions they can forward the requests. For example, node 1 requests node 3 to set up pairwise keys with nodes 4 and 5, and then forwards the pairwise key request $\{6\}$ to node 4 and request $\{8\}$ to node 5. To achieve this goal, node 1 simply generates right node *id* sequence for node 3 to recognize. In the given example, node 3 recognizes that its Q-neighbors in the request are 4 and 5; the *id* 6 follows 4, and hence is sent to node 4 along with the key k_{14} ; the *id* 8 follows 5, and hence is sent to node 5 with the key k_{15} . In order to generate right request sequence for every Q-neighbor, $\{\{2, 7\} \{3, \{\{4,6\}\{5,8\}\}\}\}$, node 1 can simply perform a depth first search on its shortest path tree constructed from the *key graph*. After generating the sequence, node 1 sends $\{7\}$ to node 2 and $\{\{4,6\}\{5,8\}\}$ to node 3. Node 3 then forwards the requests $\{6\}$ and $\{8\}$ to nodes 4 and 5 respectively. After node 4 and 5 receive the requests, node 4 generates the response $E_{k_{14}}\{k_{16}\}$, $E_{k_{46}}\{k_{16}\}$ and sends to node 1, 6 respectively; node 5 generates the response $E_{k_{15}}\{k_{18}\}$, $E_{k_{58}}\{k_{18}\}$ and sends to node 1, 8 respectively. Following the same procedure, node 2 helps node 1 and 7 to set up a pairwise key. Note that for each pair of nodes, both nodes can initiate the PKE process. For example, node 1 and node 7 can both send out requests for PKE. Since the intermediate node 2 does not maintain the state of previous requests, node 2 may send node 1 and node 7 two responses each. If a sensor receives two responses, it just simply *xor* the received keys to derive a new pairwise key.

2.4 Message Format

In this section, we present a message format that involves three preliminary functions: addressing, authenticity, and data confidentiality. The message format is shown in Figure 3.

The message format is proposed based on the com-

¹A *key path* between node A and B is defined as a sequence of nodes $A, N_1, N_2, \dots, N_i, B$, such that, each pair of nodes $(A, N_1), (N_1, N_2), \dots, (N_{i-1}, N_i), (N_i, B)$ has at least one shared key after the key discovery phase. The *length* of the *key path* is the number of pairs of nodes in it.

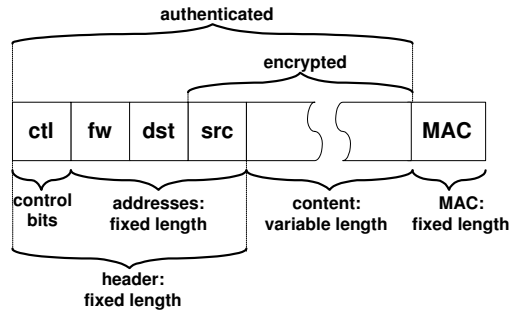


Figure 3: Message format

munication involved in pairwise key setup requests and responses. Both requests and responses should be included in a formatted message. The message includes a *header*, *content*, and message *authentication code* (MAC). The *header* and MAC have fixed length. To find the communication overhead due to the fixed part, of the message, we can multiply the total length of *header* and MAC by the number of messages transmitted during the key setup phase. The *header* contains *control bits* and at least three address fields: *src* specifies the requestor, *dst* specifies the next hop of the message, and *fw* specifies the forwarding node *id*. Note that, when the source initiates the first request message, the fields *fw* and *src* should both be set to the initiator's *id*. During the entire process of pairwise key setup, *src* always remains the same, fields *fw* and *dst* change in transit on every hop. At each hop, a node uses *src* and the node list within the *contents* to identify the pairwise keys *src* has requested.

We define three type of messages. As shown in Figure 2, the request message (*Type 1*): pairwise key setup request (for example, messages $\langle 1 \rangle$ and $\langle 2 \rangle$); response message (*Type 2*): pairwise key setup response, included the encrypted pairwise key (for example, messages $\langle 3 \rangle$, $\langle 4 \rangle$, and $\langle 7 \rangle - \langle 12 \rangle$); responses&request message (*Type 3*). When the number of forwarding hops is greater than or equal to 3, intermediate nodes on the *key path* send the messages that include both the response to pairwise key request from previous hop and the requests forwarded to the next hop (for example, messages $\langle 5 \rangle$ and $\langle 6 \rangle$).

The message is authenticated using the pairwise key that has already been set up. To prevent eavesdropping, *src id* and *contents* are encrypted using the pairwise key.

3 Performance Analysis

In this section, we present performance analysis based on the number of messages transmitted during

the PKE procedure, the energy consumption due to the transmission, and security issues due to the PKE procedure. Our analysis is based on the following probabilities introduced in [1]: (1) the probability $p_r(h)$ that two sensors can set up pairwise keys with exact h hops; (2) the probability $p_r(\leq H)$ that two sensors can set up pairwise keys within H hops, where H is the maximum number of hops allowed to establish pairwise keys; and (3) the probability $p_{con}(H)$ that a sensor can set up pairwise keys with all its neighbors within H hops.

3.1 Communication Overhead

We assume that all sensors behave correctly and the communication overhead due to retransmissions and lost messages is out of scope of this paper. In order to reduce the communication overhead, a sensor always prefers the shortest paths (with minimal number of hops) to set up pairwise keys with its R -neighbors. If there exists multiple shortest paths to an R -neighbor, one of them is randomly picked. We use n_h to represent the average number of nodes on hop h and n' to represent the average number of neighbors of a node. A node on hop h may or may not help to reach nodes at hop $h+1$. Thus, the average number of nodes that can be reached with exactly h hops is given as:

$$n_h = n' p_r(h) \quad (1)$$

In the example shown in Figure 2, node 1 sends two requests, messages $\langle 1 \rangle$ and $\langle 2 \rangle$ to set up pairwise keys with nodes 4, 5, 6, 7, and 8. The intermediate nodes (node 2 and node 3) help to setup the pairwise keys. For each pairwise key, the intermediate node will send either two *Type 2* messages, or, one *Type 2* message and one *Type 3* message. Thus, the number of transmitted keys is twice the number of nodes that can be reached with 2 or more hops from the source node 1 (10 messages $\langle 3 \rangle$ - $\langle 12 \rangle$ in the example). We use N_{key} to represent the number of transmitted pairwise keys during the PKE phase invoked by a sensor. Then we have:

$$N_{key} = 2 \sum_{h=2}^{n'} n_h \quad (2)$$

The number of keys counted by (2) includes the number of keys sent back to the source and the number of keys sent to the destination. The key messages sent back to source are always *Type 2* messages and the key messages sent to the destination can be either *Type 2* or *Type 3*. If the destination node is a leaf node in the *shortest path* tree of the source node, the key message is *Type 2* message; it is *Type 3* otherwise.

We now compute the average number of *Type 1* and *Type 3* messages (we jointly call these two types of

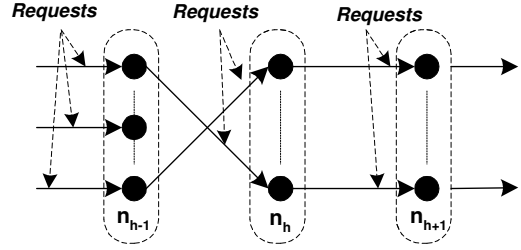


Figure 4: Pairwise key requests

messages as *requests*) that will be involved in the pairwise key setup phase. After the key discovery phase, the pairwise key setup phase is needed for sensor i if the set of R -neighbors within h hops of i is non-empty. The pairwise key(s) are established with R -neighbor(s) via Q -neighbor(s). The average number of R -neighbor(s) that can be reached via exactly h hops is given in (1). We use N_h to represent the number of *requests* that are sent to nodes on hop h . We illustrate this using Figure 4. A *request* is sent from a node on hop $h-1$ to a node on hop h for setup of pairwise keys with nodes that can be reached with $h+1$ or more hops. Whether or not a node on hop h can serve as an intermediate node between node on hop $h-1$ and node(s) on hop $h+1$ is a selection problem. A node on hop h is said to be selected if a node on hop $h-1$ selects it as the intermediate node to reach at least one of the nodes on hop $h+1$. If a node on hop h is selected, the node on hop $h-1$ sends a *request* message to it. We now solve the selection problem as follows: the probability that a node on hop h is not selected as the intermediate node to a node on hop $h+1$ is $1 - 1/n_h$; the probability that a node on hop h is not selected as the intermediate node to any of the nodes on hop $h+1$ is $(1 - 1/n_h)^{n_{h+1}}$; the probability that a node on hop h is selected as the intermediate node to at least one of nodes on hop $h+1$ is $1 - (1 - 1/n_h)^{n_{h+1}}$; the probability that at least one node on hop h is selected as the intermediate node to nodes on hop $h+1$ is $1 - (1 - 1/n_h)^{n_h n_{h+1}}$; finally, the probability (P_h) that a node on hop h is selected as the intermediate node to at least one node on hop $h+1$ given that at least one of nodes on hop h is selected is $P_h = (1 - (1 - 1/n_h)^{n_{h+1}})(1 - (1 - 1/n_h)^{n_h n_{h+1}})$. Then we have expected number of request messages on hop h as:

$$N_h = P_h n_h \quad (3)$$

Thus, the total number of request messages invoked

by a sensor is given as:

$$N_{request} = \sum_{h=1}^{n'-1} N_h$$

For $2 \leq h \leq n'$, the number of W-neighbors of a sensor that can be reached within h hops, is given by $n'p_r(\leq h)$. The number of *ids* transmitted from hop h is n_h less than that transmitted from hop $h-1$. Thus, on average, the total number of *ids* in the request messages transmitted during the PKE phase for one node is given as:

$$N_{request}^{id} = \sum_{h=1}^{H-1} \left(n'p_r(\leq H) - \sum_{k=1}^h n_k \right) \quad (4)$$

where H is the maximum number of hops allowed to establish pairwise key and $H \leq n'$.

The total number of messages transmitted during the PKE is:

$$N_{message} = N_{key} + N_1 \quad (5)$$

N_1 is the average number of nodes on hop 1 that help to set up pairwise keys. It may be noted that N_1 is also the number of *Type 1* messages sent by a sensor during the PKE phase. N_{key} includes both *Type 2* and *Type 3* messages.

On average, the communication overhead (C) invoked by a sensor during the PKE phase is given as:

$$\begin{aligned} C &= (\text{number of } id\text{s in the requests}) \times S_{id} \\ &+ (\text{number of keys}) \times S_{key} \\ &+ (\text{number of headers and MACs}) \times S_{header\&MAC} \\ &= N_{request}^{id} S_{id} + N_{key} S_{key} \\ &+ N_{message} S_{header\&MAC} \end{aligned} \quad (6)$$

where S_{key} is the size of a encrypted pairwise key, $S_{header\&MAC}$ is the total size of the *header* and MACs for each message, and S_{id} is the size of a node *id*.

In Figure 5, we compute the communication overhead invoked by a sensor using (2) and (3). We analyze four scenarios for a sensor with number of neighbors (n') equal to 10, 30, 50, and 70. Using $p_r(h)$, we derive $p_1 = p_r(1)$ that allows a node to reach 99.999% of its neighbors ($p_r(\leq H) \geq 0.99999$). The derived values of p_1 for the four scenarios are 0.9973, 0.649, 0.446, and 0.34, respectively. For small size of neighborhood, for example, $n' = 10 \sim 30$, a high percentage of connected neighbors requires large p_1 . We notice that the majority of the communication overhead is distributed within 2 hops when $p_r(\leq H) \geq 0.99999$. When the percentage of connected neighbors is greater than 50%,

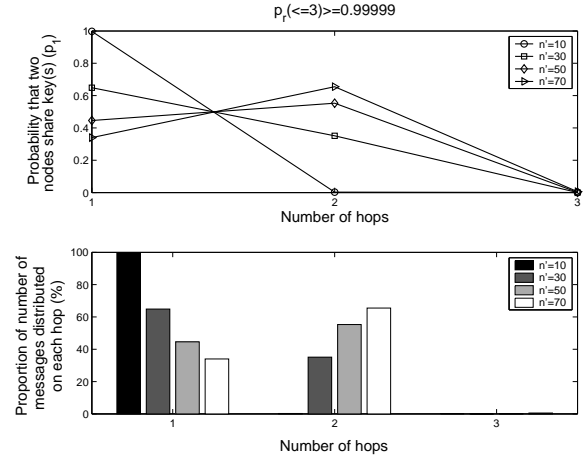


Figure 5: Sensor network key establishment communication overhead distribution for $p_r(\leq 4) \geq 0.99999$

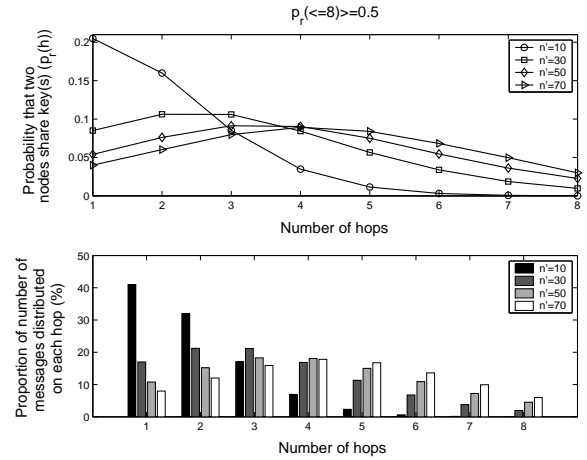


Figure 6: Sensor network key establishment communication overhead distribution for $p_r(\leq 8) \geq 0.5$

the majority communication overhead is spread out up to 8 hops (shown in Figure 6). Thus, with the decreases in the probability $p_r(\leq H)$, the total communication overhead is distributed on more hops.

3.2 Security Analysis

Many kinds of attacks can be launched on sensor networks. Specifically, such attacks can disrupt the PKE procedure; for example, wormhole attack, sink-hole attack, selective forwarding, Sybil attacks, and so on. [9]. In this paper, we do not address how to guard against these types of attacks during PKE procedure. In this section, we analyze the security vulnerabilities due to the PKE procedure. In particular, our focus is to study the number of keys that can be compromised during the PKE procedure due to a malicious node in a

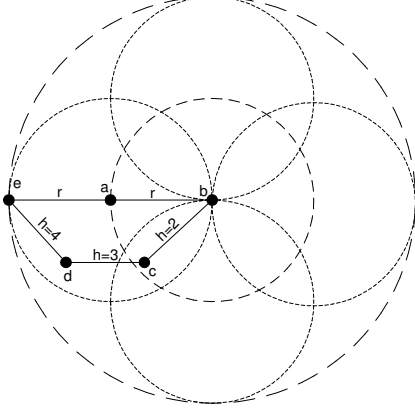


Figure 7: Possible locations of source and destination nodes for a pairwise key established via a malicious node.

neighborhood and the area in which the compromised keys will be used by uncompromised nodes. Here, we assume a malicious node behaves exactly like a normal node. We compute the number of pairwise keys that are set up via the malicious node.

In our PKE protocol, a malicious node may be located on *key paths* between nodes. Any pairwise key set up via this node is considered as a compromised key. First, we study the area in which the compromised key(s) will be used between two uncompromised nodes. Let us say that node b is malicious. Figure 7 shows the possible area in which the compromised keys will be used in the system. When the length of *key path* is less than 3, the compromised key will be used by two uncompromised nodes both located within the circular area with radius r and node b as the center. When the length of the *key path* is greater than or equal to 3, source nodes should be located within the circular area with radius r and node b as the center (for example, nodes a and c); the destination nodes should be located within the circular area with radius $2r$ and node b as the center (for example, nodes a , c , d , and e .)

From (1), a node can connect to n_h number of nodes with exactly h hops. The probability that the malicious node is one of the h -hop nodes is $p_r(h)$. Now, the probability that an attacker is one of h -hop nodes and helps to set up pairwise keys is $p_r(h)(N_h/n_h)$, where N_h is derived from (3). On hop h , the average number of pairwise key requests processed by a malicious node is $(p_r(\leq H)n' - \sum_{k=1}^h n_k)/N_h$, where H is the maximum number of hops allowed to set up pairwise keys in the system. Thus, we derive the average number of pairwise keys that an attacker can compromise on hop

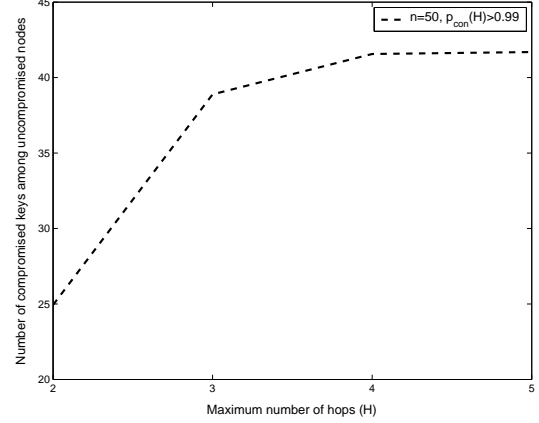


Figure 8: The average number of pairwise keys compromised by an attacker during the PKE phase ($n' = 50$, $p_{con}(H) > 0.99$; $H = 2, p_1 = 0.501$, $H = 3, p_1 = 0.286$, $H = 4, p_1 = 0.264$, and $H = 5, p_1 = 0.263$).

h as:

$$p_r(h) \left(\frac{N_h}{n_h} \right) \left(\frac{p_r(\leq H)n' - \sum_{k=1}^h n_k}{N_h} \right) = p_r(\leq H) - p_r(\leq h)$$

Thus, the average number of pairwise keys that a malicious node can help an uncompromised node to set up is given as:

$$(H-1)p_r(\leq H) - \sum_{k=1}^{H-1} p_r(\leq k)$$

Finally, because each sensor has n' neighbors, we derive the average of total number of compromised pairwise keys (N_{comp}) that can be set up via a malicious node:

$$N_{comp} = n' \left[(H-1)p_r(\leq H) - \sum_{k=1}^{H-1} p_r(\leq k) \right]. \quad (7)$$

(7) shows the average number of keys that can be compromised due to one malicious node during the PKE procedure. Figure 8, shows the average number of pairwise keys compromised by a malicious node during the PKE phase. In this figure, we analyze the scenarios with $n' = 50$, $p_{con}(H) > 0.99$, and $2 \leq H \leq 5$. The derived values of p_1 for $H = 2, 3, 4, 5$ to satisfy the probability $p_{con}(H) > 0.99$ are 0.501, 0.286, 0.264, and 0.263, respectively. For $H \geq 4$, with increase in the maximum number of hops, the number of compromised keys increase at relatively slower rate.

4 Conclusion

In this paper, we propose an energy-efficient PKE protocol to reduce the communication overhead involved in pairwise key establishment for large-scale sensor networks. Using the proposed PKE protocol, each sensor selects *key paths* to set up pairwise keys. We then use the model proposed in [1] to analyze its communication overhead, and security vulnerabilities.

The modelling of PKE for wireless sensor system is in its preliminary stage. We aim to propose a multiple *key paths* PKE protocol in our future work.

References

- [1] D. Huang, M. Mehta, D. Medhi, and L. Harn, "Modeling pairwise key establishment for random key predistribution in large-scale sensor networks," *submitted for publication and available at http://conrel.sice.umkc.edu/HRP/modelling_pairwise_key_establishment_schemes-v2.pdf*, 2004.
- [2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, pp. 102–114, August 2002.
- [3] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of 9th ACM Conference on Computer and Communication Security (CCS-02)*, November 2002, pp. 41–47.
- [4] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of 2003 Symposium on Security and Privacy*. Los Alamitos, CA: IEEE Computer Society, 11–14 2003, pp. 197–215.
- [5] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, October 2003, pp. 52–61.
- [6] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, October 2003, pp. 42–51.
- [7] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Establishing pair-wise keys for secure communication in ad hoc networks: A probabilistic approach," in *Proceedings of 11th IEEE International Conference on Network Protocols (ICNP)*, November 2003.
- [8] D. W. Carman, P. S. Kruus, and B. J. Matt, "Constraints and approaches for distributed sensor network security," NAI Lab, Tech. Rep., September 2000.
- [9] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, vol. 1, no. 2–3, pp. 293–315, September 2003.