A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks[*]

Wenliang Du
Systems Assurance Institute
Department of Electrical Engineering and Computer Science
Syracuse University
Syracuse, NY 13244-1240, USA.
wedu@ecs.syr.edu

Jing Deng
Department of Electrical Engineering and Computer Science
Syracuse University
Syracuse, NY 13244-1240, USA.
jdeng01@ecs.syr.edu

Yunghsiang S. Han[†]
Department of Computer Science and Information Engineering
National Chi Nan University
Taiwan, R.O.C.
yshan@csie.ncnu.edu.tw

Pramod K. Varshney
Department of Electrical Engineering and Computer Science
Syracuse University
Syracuse, NY 13244-1240, USA.
varshney@ecs.syr.edu

ABSTRACT

To achieve security in wireless sensor networks, it is important to be able to encrypt and authenticate messages sent among sensor nodes. Keys for encryption and authentication purposes must be agreed upon by communicating nodes. Due to resource constraints, achieving such key agreement in wireless sensor networks is nontrivial. Many key agreement schemes used in general networks, such as Diffie-Hellman and public-key based schemes, are not suitable for wireless sensor networks. Pre-distribution of secret keys for all pairs of nodes is not viable due to the large amount of memory used when the network size is large. To solve the key pre-distribution problem, two elegant key pre-distribution approaches have been proposed recently [11, 7].

In this paper, we propose a new key pre-distribution scheme, which substantially improves the resilience of the network compared to the existing schemes. Our scheme exhibits a nice threshold property: when the number of compromised nodes is less than the threshold, the probability that any nodes other than these compromised nodes is affected is close to zero. This desirable property lowers the initial payoff of smaller scale network breaches to an adversary, and makes it necessary for the adversary to attack a significant proportion of the network. We also present an in depth analysis of our scheme in terms of network resilience and associated overhead.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Security, Design, Algorithms

Keywords

Wireless sensor networks, key pre-distribution, security

1. INTRODUCTION

Recent advances in electronic and computer technologies have paved the way for the proliferation of wireless sensor networks (WSN). Sensor networks usually consist of a large number of ultra-small autonomous devices. Each device, called a sensor node, is battery powered and equipped with integrated sensors, data processing capabilities, and short-range radio communications. In typical application scenarios, sensor nodes are spread randomly over the terrain under scrutiny and collect sensor data. Examples of sensor network projects include SmartDust [12] and WINS [1].

Sensor networks are being deployed for a wide variety of applications [2], including military sensing and tracking, environment monitoring, patient monitoring and tracking, smart environments, etc. When sensor networks are deployed in a hostile environment, security becomes extremely important, as they are prone to different types of malicious attacks. For example, an adversary can easily listen to the traffic, impersonate one of the network nodes, or intentionally provide misleading information to other nodes. To

[*]This work was supported in part by Grant ISS-0219560 from the National Science Foundation, by the SUPRIA program of the CASE Center at Syracuse University, and by the National Science Council of Taiwan, R.O.C., under grants NSC 90-2213-E-260-007 and NSC 91-2213-E-260-021.

[†]Han's work was completed during his visit to the CASE Center and Department of Electrical Engineering and Computer Science at Syracuse University, USA.

provide security, communication should be encrypted and authenticated. The open problem is how to bootstrap secure communications between sensor nodes, i.e. how to set up secret keys between communicating nodes?

This problem is known as the *key agreement* problem, which has been widely studied in general network environments. There are three types of general key agreement schemes: trusted-server scheme, self-enforcing scheme, and key pre-distribution scheme. The *trusted-server* scheme depends on a trusted server for key agreement between nodes, e.g., Kerberos [15]. This type of scheme is not suitable for sensor networks because there is no trusted infrastructure in sensor networks. The *self-enforcing* scheme depends on asymmetric cryptography, such as key agreement using public key certificates. However, limited computation and energy resources of sensor nodes often make it undesirable to use public key algorithms, such as Diffie-Hellman key agreement [8] or RSA [18], as pointed out in [16]. The third type of key agreement scheme is key *pre-distribution*, where key information is distributed among all sensor nodes prior to deployment. If we know which nodes will be in the same neighborhood before deployment, keys can be decided *a priori*. However, most sensor network deployments are random; thus, such *a priori* knowledge does not exist.

There exist a number of key pre-distribution schemes which do not rely on *a priori* deployment knowledge. A naive solution is to let all the nodes carry a *master* secret key. Any pair of nodes can use this global master secret key to achieve key agreement and obtain a new pairwise key. This scheme does not exhibit desirable network resilience: if one node is compromised, the security of the entire sensor network will be compromised. Some existing studies suggest storing the master key in tamper-resistant hardware to reduce the risk, but this increases the cost and energy consumption of each sensor. Furthermore, tamper-resistant hardware might not always be safe [3]. Another key pre-distribution scheme is to let each sensor carry $N - 1$ secret pairwise keys, each of which is known only to this sensor and one of the other $N - 1$ sensors (assuming N is the total number of sensors). The resilience of this scheme is perfect because a compromised node does not affect the security of other nodes; however, this scheme is impractical for sensors with an extremely limited amount of memory because N could be large. Moreover, adding new nodes to a pre-existing sensor network is difficult because the existing nodes do not have the new nodes' keys.

Very recently Eschenauer and Gligor proposed a random key pre-distribution scheme: before deployment, each sensor node receives a random subset of keys from a large key pool; to agree on a key for communication, two nodes find one common key within their subsets and use that key as their shared secret key [11]. Based on this scheme, Chan, Perrig, and Song proposed a q-composite random key pre-distribution scheme, which increases the security of key setup such that an attacker has to compromise many more nodes to achieve a high probability of compromising communication [7]. The difference between the q-composite scheme and the scheme in [11] is that q common keys ($q \geq 1$), instead of just a single one, are needed to establish secure communication between a pair of nodes. It is shown that by increasing the value of q network resilience against node capture is improved [7].

1.1 Main Contributions of Our Scheme

In this paper, we propose a new key pre-distribution scheme. The main contributions of this paper are as follows:

1. Substantially improved network resilience against node capture over existing schemes.
2. Pairwise keys that enable authentication.
3. Thorough theoretical analysis of security, and communication and computation overhead analysis.

Our scheme builds on Blom's key pre-distribution scheme [4] and combines the random key pre-distribution method with it. Our results show that the resilience of our scheme is substantially better than Blom's scheme as well as other random key pre-distribution schemes. In [4], Blom proposed a key pre-distribution scheme that allows *any* pair of nodes to find a secret pairwise key between them. Compared to the $(N - 1)$-pairwise-key pre-distribution scheme, Blom's scheme only uses $\lambda+1$ memory spaces with λ much smaller than N. The tradeoff is that, unlike the $(N - 1)$-pairwise-key scheme, Blom's scheme is not perfectly resilient against node capture. Instead it has the following λ-secure property: *as long as an adversary compromises less than or equal to λ nodes, uncompromised nodes are perfectly secure; when an adversary compromises more than λ nodes, all pairwise keys of the entire network are compromised.*

The threshold λ can be treated as a security parameter in that selection of a larger λ leads to a more secure network. This threshold property of Blom's scheme is a desirable feature because an adversary needs to attack a significant fraction of the network in order to achieve high payoff. However, λ also determines the amount of memory to store key information, as increasing λ leads to higher memory usage. *The goal of our scheme is to increase network's resilience against node capture without using more memory.*

Blom's scheme uses *one* key space for all nodes to make sure that any pair can compute its pairwise key in this key space. Motivated by the random key pre-distribution schemes presented in [11, 7], we propose a new scheme using *multiple* key spaces: we first construct ω spaces using Blom's scheme, and each sensor node carries key information from τ ($2 \leq \tau < \omega$) randomly selected key spaces. According to Blom's scheme, if two nodes carry key information from a common space, they can compute their pairwise key from the information; when two nodes do not carry key information from a common space, they can conduct key agreement via other nodes which share pairwise keys with them. Our analysis has shown that using the same amount of memory, our new scheme is substantially more resilient than Blom's scheme and other key pre-distribution schemes.

To further improve the resilience, we also develop a two-hop-neighbor key pre-distribution scheme. The idea is to let the direct neighbor forward the message from a sender, such that nodes that are two hops away from the sender can also receive the message. The nodes that are two hops away are known as two-hop neighbors. Treating two-hop neighbors as "direct" neighbors, the number of neighbors of each sender increases fourfold. The consequence is that the resilience threshold can be improved as well. Our results show that under certain conditions, the threshold can be improved to four times as much as that of our first scheme.

The rest of the paper is organized as follows. Section 2 describes how our building block, the original Blom's method, works. Then we describe our key pre-distribution scheme in Section 3. Section 4 shows the resilience of our scheme against node capture. It also compares our scheme with existing key pre-distribution schemes. Section 5 presents the communication and computation overheads of our scheme. Section 6 describes our two-hop-neighbor key pre-distribution scheme. Finally, we provide some concluding remarks in Section 7.

1.2 Other Related Work

The Eschenauer-Gligor scheme [11] and the Chan-Perrig-Song scheme [7] have been reviewed earlier in this section. Detailed comparisons with these two schemes will be given in Section 4. Some other related work is discussed next.

Du et al. proposed a method to improve the Eschenauer-Gligor scheme using *a priori* deployment knowledge [9]. This method can also be used to further improve other random key pre-distribution schemes, such as the Chan-Perrig-Song scheme and the scheme presented in this paper.

Blundo et al. proposed several schemes which allow any group of t parties to compute a common key while being secure against collusion between some of them [5]. These schemes focus on saving communication costs while memory constraints are not placed on group members. When $t = 2$, one of these schemes is actually a special case of Blom's scheme [4]. A modified version of Blom's scheme will be reviewed in Section 2. Compared to Blom's scheme, our scheme is more resilient and more memory-efficient.

Perrig et al. proposed SPINS, a security architecture specifically designed for sensor networks [16]. In SPINS, each sensor node shares a secret key with the base station. Two sensor nodes cannot directly establish a secret key. However, they can use the base station as a trusted third party to set up the secret key.

2. BACKGROUND: BLOM'S KEY PRE-DISTRIBUTION SCHEME

Blom proposed a key pre-distribution method that allows any pair of nodes in a network to be able to find a pairwise secret key [4]. As long as no more than λ nodes are compromised, the network is perfectly secure (this is called the λ-secure property). We briefly describe how Blom's λ-secure key pre-distribution system works. Blom's scheme is not developed for sensor networks, so in the following description, we have made some slight modifications to the original scheme to make it suitable for sensor networks.

During the pre-deployment phase, the base station first constructs a $(\lambda+1) \times N$ matrix G over a finite field $GF(q)$, where N is the size of the network. G is considered as public information; any sensor can know the contents of G, and even adversaries are allowed to know G. Then the base station creates a random $(\lambda+1)\times(\lambda+1)$ symmetric matrix D over $GF(q)$, and computes an $N \times (\lambda+1)$ matrix $A = (D \cdot G)^T$, where $(D \cdot G)^T$ is the transpose of $D \cdot G$. Matrix D needs to be kept secret, and should not be disclosed to adversaries or any sensor node (although, as will be discussed later, one row of $(D \cdot G)^T$ will be disclosed to each sensor node). Because D is symmetric, it is easy to see:

$$\begin{aligned} A \cdot G &= (D \cdot G)^T \cdot G = G^T \cdot D^T \cdot G = G^T \cdot D \cdot G \\ &= (A \cdot G)^T. \end{aligned}$$

This means that $A \cdot G$ is a symmetric matrix. If we let $K = A \cdot G$, we know that $K_{ij} = K_{ji}$, where K_{ij} is the element in K located in the ith row and jth column. We use K_{ij} (or K_{ji}) as the pairwise key between node i and node j. Fig. 1 illustrates how the pairwise key $K_{ij} = K_{ji}$ is generated. To carry out the above computation, nodes i and j should be able to compute K_{ij} and K_{ji}, respectively. This can be easily achieved using the following key pre-distribution scheme, for $k = 1, \ldots, N$:

1. store the kth row of matrix A at node k, and
2. store the kth column of matrix G at node k.[1]

Therefore, when nodes i and j need to find the pairwise key between them, they first exchange their columns of G, and then they can compute K_{ij} and K_{ji}, respectively, using their private rows of A. Because G is public information, its columns can be transmitted in plaintext. It has been proved in [4] that the above scheme is λ-secure if any $\lambda + 1$ columns of G are linearly independent. This λ-secure property guarantees that no nodes other than i and j can compute K_{ij} or K_{ji} if no more than λ nodes are compromised.

[1]We will show later that each sensor does not need to store the whole column, because each column can be generated from a seed.

An Example of Matrix G

We show an example of matrix G. Note that any $\lambda + 1$ columns of G must be linearly independent in order to achieve the λ-secure property. Since each pairwise key is represented by an element in the finite field $GF(q)$, if the length of pairwise keys is 64 bits, then we should choose q as the smallest prime number[2] that is larger than 2^{64}. Let s be a primitive element of $GF(q)$ and $N < q$. That is, each nonzero element in $GF(q)$ can be represented by some power of s, namely s^i for some $0 < i \leq q-1$. A feasible G can be designed as follows [13]:

$$G = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ s & s^2 & s^3 & \cdots & s^N \\ s^2 & (s^2)^2 & (s^3)^2 & \cdots & (s^N)^2 \\ & & & \vdots & \\ s^\lambda & (s^2)^\lambda & (s^3)^\lambda & \cdots & (s^N)^\lambda \end{bmatrix}$$

It is well-known that $s^i \neq s^j$ if $i \neq j$ (this is a property of primitive elements). Since G is a Vandermonde matrix, it can be shown that any $\lambda + 1$ columns of G are linearly independent when $s, s^2, s^3, \ldots, s^N$ are all distinct [13]. In practice, G can be generated by the primitive element s of $GF(q)$. Therefore, when we store the kth column of G at node k, we only need to store the seed s^k at this node, and any node can regenerate the column given the seed. The issue of memory usage and computational complexity will be discussed later in the paper.

3. MULTIPLE-SPACE KEY PRE-DISTRIBUTION SCHEME

To achieve better resilience against node capture, we propose a new key pre-distribution scheme that uses Blom's method as a building block. Our idea is based on the following observations: Blom's method guarantees that any pair of nodes can find a secret key between themselves. To represent this we use concepts from graph theory and draw an edge between two nodes if and only if they can find a secret key between themselves. We will get a *complete* graph (i.e., an edge exists between all node pairs). Although full connectivity is desirable, it is not necessary. To achieve our goal of key agreement, all we need is a *connected* graph, rather than a complete graph. Our hypothesis is that *by requiring the graph to be only connected, each sensor node needs to carry less key information.*

Before we describe our proposed scheme, we define a *key space* (or *space* in short) as a tuple (D, G), where matrices D and G are as defined in Blom's scheme. We say a node picks a key space (D, G) if the node carries the secret information generated from (D, G) using Blom's scheme. Two nodes can calculate their pairwise key if they have picked a common key space.

[2]When q is a prime, all elements in $GF(q)$ can be represented by the nonnegative integers less than q. The addition and multiplication in $GF(q)$ are ordinary integer additions and multiplication modulo q. For example, if we want to multiply two elements in $GF(q)$, first we multiply them as ordinary integers and then carry out the modulo q operation.

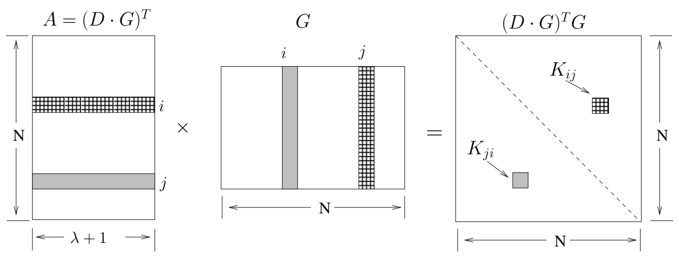


Figure 1: Generating Keys in Blom's Scheme

3.1 Key Pre-distribution Phase

During the key pre-distribution phase, we need to assign key information to each node, such that after deployment, neighboring sensor nodes can find a secret key between them. Assume that each sensor node has a unique identification, whose range is from 1 to N. We also select the security parameters τ, ω, and λ, where $2 \leq \tau < \omega$. These parameters decide the security and performance of our scheme, and will be discussed later in the paper. Our key pre-distribution phase contains the following steps:

Step 1 (Generating G matrix): We first select a primitive element from a finite field $GF(q)$, where q is the smallest prime larger than the key size, to create a generator matrix G of size $(\lambda+1) \times N$. Let $G(j)$ represent the jth column of G. We provide $G(j)$ to node j. As we have already shown in Section 2, although $G(j)$ consists of $(\lambda+1)$ elements, each sensor only needs to remember one seed (the second element of the column), which can be used to regenerate all the elements in $G(j)$. Therefore the memory usage for storing $G(j)$ at a node is just a single element. Since the seed is unique for each sensor node, it can also be used for node id.

Step 2 (Generating D matrix): We generate ω symmetric matrices $D_1, \ldots, D_\omega$ of size $(\lambda + 1) \times (\lambda + 1)$. We call each tuple $S_i = (D_i, G)$, $i = 1, \ldots, \omega$, a key space. We then compute the matrix $A_i = (D_i \cdot G)^T$. Let $A_i(j)$ represent the jth row of A_i.

Step 3 (Selecting τ spaces): We randomly select τ distinct key spaces from the ω key spaces for each node. For each space S_i selected by node j, we store the jth row of A_i (i.e. $A_i(j)$) at this node. This information is secret and should stay within the node; under no circumstance should a node send this secret information to any other node. According to Blom's scheme, two nodes can find a common secret key if they have both picked a common key space.

Since A_i is an $N \times (\lambda + 1)$ matrix, $A_i(j)$ consists of $(\lambda + 1)$ elements. Therefore, each node needs to store $(\lambda+1)\tau$ elements in its memory. Because the length of each element is the same as the length of secret keys, the memory usage of each node is $(\lambda + 1)\tau$ times the length of the key.

3.2 Key Agreement Phase

After deployment, each node needs to discover whether it shares any space with its neighbors. To do this, each node broadcasts a message containing the following information: (1) the node's id, (2) the indices of the spaces it carries, [3] and (3) the seed of the column of G it carries. [4]

Assume that nodes i and j are neighbors, and they have received the above broadcast messages. If they find out that they have a common space, e.g. S_c, they can compute their pairwise secret key using Blom's scheme: Initially node i has $A_c(i)$ and seed for $G(i)$, and node j has $A_c(j)$ and seed for $G(j)$. After exchanging the seeds, node i can regenerate $G(j)$ and node j can regenerate $G(i)$; then the pairwise secret key between nodes i and j, $K_{ij} = K_{ji}$, can be computed in the following manner by these two nodes independently:

$$K_{ij} = K_{ji} = A_c(i) \cdot G(j) = A_c(j) \cdot G(i).$$

After secret keys with neighbors are set up, the entire sensor network forms the following *Key-Sharing Graph*:

DEFINITION 3.1. *(Key-Sharing Graph) Let V represent all the nodes in the sensor network. A Key-Sharing graph $G_{ks}(V, E)$ is constructed in the following manner: For any two nodes i and j in V, there exists an edge between them if and only if (1) nodes i and j have at least one common key space, and (2) nodes i and j can reach each other within the wireless transmission range.*

We now show how two neighboring nodes, i and j, who do not share a common key space could still come up with a pairwise secret key between them. The idea is to use the secure channels that have already been established in the key-sharing graph G_{ks}: as long as G_{ks} is connected, two neighboring nodes i and j can always find a path in G_{ks} from i to j. Assume that the path is i, v_1, $\ldots$, v_t, j. To find a common secret key between i and j, i first generates a random key K. Then i sends the key to v_1 using the secure link between i and v_1; v_1 sends the key to v_2 using the secure link between v_1 and v_2, and so on until j receives the key from v_t. Nodes i and j use this secret key K as their pairwise key. Because the key is always forwarded over a secure link, no nodes beyond this path can find out the key.

3.3 Computing ω, τ, and Memory Usage

As we have just shown, to make it possible for any pair of nodes to be able to find a secret key between them, the key sharing graph $G_{ks}(V, E)$ needs to be *connected*. Given the size and the density of a network, how can we select the values for ω and τ, s.t., the graph G_{ks} is connected with high probability? We use the following three-step approach, which is adapted from [11].

Step 1: Computing Required Local Connectivity. Let P_c be the probability that the key-sharing graph is connected. We call it *global connectivity*. We use *local connectivity* to refer to the probability of two neighboring nodes sharing at least one space (i.e. they can find a common key between them). The global connectivity and the local connectivity are related: to achieve a desired global connectivity P_c, the local connectivity must be higher than a certain value; we call this value the *required local connectivity*, denoted by $p_{required}$.

[3] If we are concerned about disclosing the indices of the spaces each node carries, we can use the challenge-response technique to avoid sending the indices [7].

[4] We could also let node id be the same as the seed.

Using connectivity theory in a random-graph by Erdős and Rényi [10], we can obtain the necessary expected node degree d (i.e., the average number of edges connected to each node) for a network of size N when N is large in order to achieve a given global connectivity, P_c:

$$d = \frac{(N-1)}{N}\left[\ln(N) - \ln(-\ln(P_c))\right]. \tag{1}$$

For a given density of sensor network deployment, let n be the expected number of neighbors within wireless communication range of a node. Since the expected node degree must be at least d as calculated above, the required local connectivity $p_{required}$ can be estimated as:

$$p_{required} = \frac{d}{n}. \tag{2}$$

Step 2: Computing Actual Local Connectivity. After we have selected values for ω and τ, the actual local connectivity is determined by these values. We use p_{actual} to represent the actual local connectivity, namely p_{actual} is the actual probability of any two neighboring nodes sharing at least one space (i.e. they can find a common key between them). Since $p_{actual} = 1 - \Pr(\text{two nodes do not share any space})$,

$$p_{actual} = 1 - \frac{\binom{\omega}{\tau}\binom{\omega-\tau}{\tau}}{\binom{\omega}{\tau}^2} = 1 - \frac{((\omega-\tau)!)^2}{(\omega-2\tau)!\omega!}. \tag{3}$$

The values of p_{actual} have been plotted in Fig. 2 when ω varies from τ to 100 and $\tau = 2, 4, 6, 8$. For example, one can see that, when $\tau = 4$, the largest ω that we can choose while achieving the local connectivity $p_{actual} \geq 0.5$ is 25.

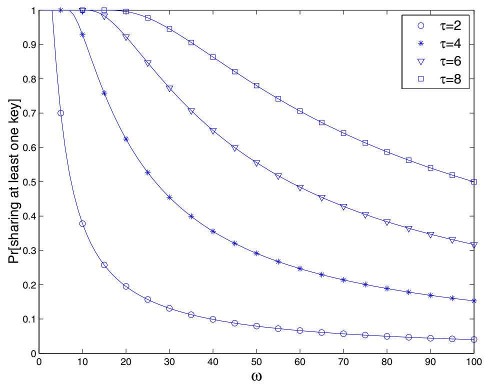


Figure 2: Probability of sharing at least one key when two nodes each randomly chooses τ spaces from ω spaces.

The collection of sets of spaces assigned to each sensor form a probabilistic quorum system [14]: the desire is that every two sensors have a space in common with high probability. Furthermore, it can be shown that if $\tau \geq \sqrt{\ln \frac{1}{1-p_{actual}}}\sqrt{\omega}$, then the probability of intersection is at least p_{actual}; this has the similar property to the birthday paradox. For example, when $\tau \geq \sqrt{\ln 2}\sqrt{\omega}$, the probability of intersection is at least $1/2$. This can explain the behavior of Fig. 2.

Step 3: Computing ω and τ. Knowing the required local connectivity $p_{required}$ and the actual local connectivity p_{actual}, in order to achieve the desired global connectivity P_c, we should have $p_{actual} \geq p_{required}$,

$$1 - \frac{((\omega-\tau)!)^2}{(\omega-2\tau)!\omega!} \geq \frac{(N-1)}{nN}\left[\ln(N) - \ln(-\ln(P_c))\right]. \tag{4}$$

Therefore, in order to achieve a certain P_c for a network of size N and the expected number of neighbors for each node being n, we just need to find values of ω and τ, such that Inequality (4) is satisfied.

Step 4: Computing memory usage. According to Blom's scheme, a node needs to store a row from an $N \times (\lambda+1)$ matrix $(D \cdot G)^T$; therefore, for each selected space, a node needs to carry $\lambda + 1$ elements; Hence the total memory usage m for each node is:

$$m = (\lambda+1)\tau. \tag{5}$$

4. SECURITY ANALYSIS

We evaluate the multiple-space key pre-distribution scheme in terms of its resilience against node capture. Our evaluation is based on two metrics: (1) When x nodes are captured, what is the probability that at least one key space is broken? As we know, because of the λ-secure property of our scheme, to break a key space, an adversary needs to capture $\lambda+1$ nodes that contain this key space's information; otherwise, the key space is still perfectly secure. This analysis shows when the network starts to become insecure. (2) When x nodes are captured, what fraction of the additional communication (i.e. communication among uncaptured nodes) also becomes compromised? This analysis shows how much payoff an adversary can gain after capturing a certain number of nodes.

4.1 Probability of At Least One Space Being Broken

We define the unit of memory size as the size of a secret key (e.g. 64 bits). According to Blom's scheme, if a space is λ-secure, each node needs to use memory of size $\lambda + 1$ to store the space information. Therefore, if the memory usage is m and each node needs to carry τ spaces, then the value of λ should be $\lfloor \frac{m}{\tau} \rfloor - 1$. In the following analysis, we choose $\lambda = \lfloor \frac{m}{\tau} \rfloor - 1$.

Let $\mathcal{S}_i$ be the event that space S_i is broken, where $i = 1, \ldots, \omega$, and $\mathcal{C}_x$ be the event that x nodes are compromised in the network. Furthermore, let $\mathcal{S}_i \cup \mathcal{S}_j$ be the joint event that either space S_i or space S_j, or both, is broken and $\theta = \frac{\tau}{\omega}$. Hence, we have

$$\Pr(\text{at least one space is broken} \mid \mathcal{C}_x) = \Pr(\mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_\omega \mid \mathcal{C}_x).$$

According to the Union Bound,

$$\Pr(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_\omega \mid \mathcal{C}_x) \leq \sum_{i=1}^{\omega} \Pr(\mathcal{S}_i \mid \mathcal{C}_x).$$

Due to the fact that each key space is broken with equal probability,

$$\sum_{i=1}^{\omega} \Pr(\mathcal{S}_i \mid \mathcal{C}_x) = \omega \Pr(\mathcal{S}_1 \mid \mathcal{C}_x).$$

Therefore,

$$\begin{aligned} &\Pr(\text{at least one space is broken} \mid \mathcal{C}_x) \\ \leq\ &\sum_{i=1}^{\omega} \Pr(\mathcal{S}_i \mid \mathcal{C}_x) = \omega \Pr(\mathcal{S}_1 \mid \mathcal{C}_x). \end{aligned} \tag{6}$$

We now need to calculate $\Pr(\mathcal{S}_1 \mid \mathcal{C}_x)$, the probability of space S_1 being compromised when x nodes are compromised. Because each node carries information from τ spaces, the probability that each compromised node carries information about S_1 is $\theta = \frac{\tau}{\omega}$. Therefore, after x nodes are compromised, the probability that exactly j of these x nodes contain information about S_1 is $\binom{x}{j}\theta^j(1-\theta)^{x-j}$. Since space S_1 can only be broken after at least $\lambda+1$ nodes are compromised, we have the following result:

$$\Pr(\mathcal{S}_1 \mid \mathcal{C}_x) = \sum_{j=\lambda+1}^{x} \binom{x}{j} \theta^j (1-\theta)^{x-j}. \tag{7}$$

Combining Inequality (6) and Equation (7), we have the following upper bound:

$$\begin{aligned} &\Pr(\text{at least one space is broken} \mid \mathcal{C}_x) \\ \le\ & \omega \sum_{j=\lambda+1}^{x} \binom{x}{j} \theta^j (1-\theta)^{x-j} \\ =\ & \omega \sum_{j=\lambda+1}^{x} \binom{x}{j} \left(\frac{\tau}{\omega}\right)^j \left(1-\frac{\tau}{\omega}\right)^{x-j}. \end{aligned} \tag{8}$$

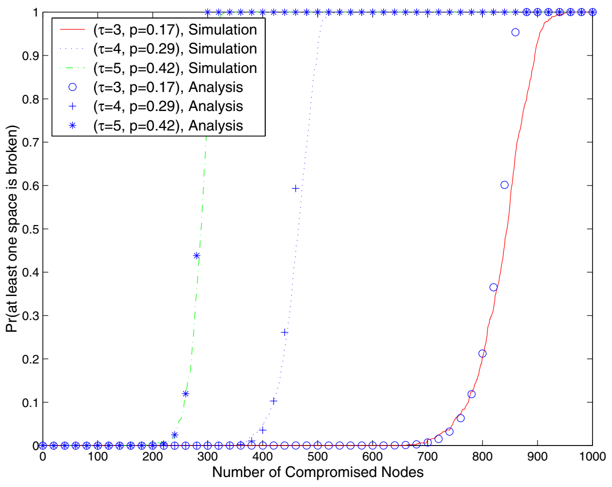


Figure 3: The probability of at least one key space being compromised by the adversary when the adversary has captured x nodes ($m = 200$, $\omega = 50$). p in the figure represents p_{actual}.

We plot both simulation and analytical results in Fig. 3. From the figure, the two results match each other closely, meaning that the union bound works quite well in the scenarios we discuss. Fig. 3 shows, for example, when the memory usage is set to 200, ω is set to 50, and τ is set to 4, the value of λ for each space is $49 = \lfloor \frac{200}{4} \rfloor - 1$, but an adversary needs to capture about 380 nodes in order to be able to break at least one key space with non-negligible probability.

Authentication Property

Due to the property of Blom's scheme, all keys generated in a space are pairwise keys. Therefore, when the space is not yet compromised, keys in this space can be used for authentication purposes. After the space is broken, adversaries can generate all the pairwise keys in that space, and keys in that space can no longer be used for authentication purposes. According to our analysis, adversaries need to compromise a significant number of nodes in order to compromise a space.

4.2 The Fraction of Network Communications that is Compromised

To understand the resilience of our key pre-distribution scheme, we need to find out how the capture of x sensor nodes by an adversary affects the rest of the network. In particular, we want to find out the fraction of additional communications (i.e., communications among uncaptured nodes) that an adversary can compromise based on the information retrieved from the x captured nodes. To compute this fraction, we first compute the probability that any one of the additional communication links is compromised after x nodes are captured. Note that we only consider the links in the key-sharing graph, and each of these links is secured using a pairwise key computed from the common key space shared by the two nodes of this link. We should also notice that after the key setup stage, two neighboring nodes can use the established secure links to agree upon another random key to secure their communication. Because this key is not generated from any key space, the security of this new random key does not directly depend on whether the key spaces are broken. However, if an adversary can record all the communications during the key setup stage, he/she can still compromise this new key after compromising the corresponding links in the key-sharing graph.

Let c be a link in the key-sharing graph between two nodes that are not compromised, and K be the communication key used for this link. Let $\mathcal{B}_i$ represent the joint event that K belongs to space S_i and space S_i is compromised. We use $K \in S_i$ to represent that "K belongs to space S_i". The probability of c being broken given x nodes are compromised is:

$$\Pr(c \text{ is broken} \mid \mathcal{C}_x) = \Pr(\mathcal{B}_1 \cup \mathcal{B}_2 \cup \cdots \cup \mathcal{B}_\omega \mid \mathcal{C}_x).$$

Since c can only use one key, events $\mathcal{B}_1, \ldots, \mathcal{B}_\omega$ are mutually exclusive. Therefore,

$$\Pr(c \text{ is broken} \mid \mathcal{C}_x) = \sum_{i=1}^{\omega} \Pr(\mathcal{B}_i \mid \mathcal{C}_x) = \omega \Pr(\mathcal{B}_1 \mid \mathcal{C}_x),$$

because all events $\mathcal{B}_i$ are equally likely. Note that

$$\Pr(\mathcal{B}_1 \mid \mathcal{C}_x) = \frac{\Pr((K \in S_1) \cap (S_1 \text{ is compromised}) \cap \mathcal{C}_x)}{\Pr(\mathcal{C}_x)}.$$

Since the event $(K \in S_1)$ is independent of the event $\mathcal{C}_x$ or the event (S_1 is compromised),

$$\begin{aligned} \Pr(\mathcal{B}_1 \mid \mathcal{C}_x) &= \frac{\Pr(K \in S_1) \cdot \Pr(S_1 \text{ is compromised } \cap \mathcal{C}_x)}{\Pr(\mathcal{C}_x)} \\ &= \Pr(K \in S_1) \cdot \Pr(S_1 \text{ is compromised} \mid \mathcal{C}_x). \end{aligned}$$

$\Pr(S_1 \text{ is compromised} \mid \mathcal{C}_x)$ can be calculated by Equation (7). The probability that K belongs to space S_1 is the probability that link c uses a key from space S_1. Since the choice of a space from ω key spaces is equally probable, we have:

$$\Pr(K \in S_1) = \Pr(\text{the link } c \text{ uses a key from space } S_1) = \frac{1}{\omega}.$$

Therefore,

$$\begin{aligned} &\Pr(c \text{ is broken} \mid \mathcal{C}_x) \\ =\ & \omega \Pr(\mathcal{B}_1 \mid \mathcal{C}_x) = \omega \cdot \frac{1}{\omega} \cdot \Pr(S_1 \text{ is compromised} \mid \mathcal{C}_x) \\ =\ & \Pr(S_1 \text{ is compromised} \mid \mathcal{C}_x) \\ =\ & \sum_{j=\lambda+1}^{x} \binom{x}{j} \left(\frac{\tau}{\omega}\right)^j \left(1-\frac{\tau}{\omega}\right)^{x-j}. \end{aligned} \tag{9}$$

Assume that there are γ secure communication links that do not involve any of the x compromised nodes. Given the probability $\Pr(c \text{ is broken} \mid \mathcal{C}_x)$, we know that the expected fraction of broken communication links among those γ links is

$$\begin{aligned} & \frac{\gamma \cdot \Pr(c \text{ is broken} \mid \mathcal{C}_x)}{\gamma} \\ = \; & \Pr(c \text{ is broken} \mid \mathcal{C}_x) \\ = \; & \Pr(S_1 \text{ is compromised} \mid \mathcal{C}_x). \end{aligned} \tag{10}$$

The above equation indicates that, given that x nodes are compromised, the fraction of the compromised secure communication links outside of those x compromised nodes is the same as the probability of one space being compromised. This can be explained quite intuitively. Since spaces are selected in an equally likely fashion during the key pre-distribution process, after x nodes are compromised, the expected number of spaces that are compromised is about $\omega \Pr(S_1 \text{ is compromised} \mid \mathcal{C}_x)$. Therefore, the fraction of the spaces that are compromised is $\Pr(S_1 \text{ is compromised} \mid \mathcal{C}_x)$. Because keys from different spaces are evenly selected by the communication links, the fraction of communication links compromised should be the same as the fraction of the spaces compromised. Therefore, the fraction of the spaces compromised is also $\Pr(S_1 \text{ is compromised} \mid \mathcal{C}_x)$.

4.2.1 Comparison

Fig. 4 shows the comparison of our scheme (the one with solid lines) with the Chan-Perrig-Song scheme ($q = 2$, $q = 3$) and the Eschenauer-Gligor scheme ($q = 1$). The figure clearly shows the advantage of our scheme. For example, when the memory usage m is the same ($m = 200$), and $p_{actual} = 0.33$, with both Chan-Perrig-Song and Eschenauer-Gligor schemes, an adversary only needs to compromise less than 100 nodes in order to compromise 10% of the rest of the secure links, whereas in our scheme, the adversary needs to compromise 500 nodes. Therefore, our scheme quite substantially lowers the initial payoff to the adversary of smaller scale network breaches. Chan, Perrig, and Song also proposed a modification of their scheme using multipath key reinforcement to improve the security [7]. The same technique can also be applied to our scheme to improve the security of our scheme as well; we leave further comparison to our future work.

Regarding the original Blom's scheme, because $m = 200$, the network is perfectly secure if less than 200 nodes are compromised; the network is completely compromised when 200 nodes are compromised (p_{actual} is always equal to 1 in Blom's scheme).

4.2.2 Further Analysis

Even though Equation (9) can be used for numerical computation, it is too complicated to figure out the relationship between x, m, ω, and τ. According to the results shown in Fig. 4, there is a small range of x where the fraction of the compromised secure communication links increases exponentially with respect to x. We develop an analytical form to estimate this range. It should be noted that Equation (9) is the tail of the binomial distribution. Therefore, using the bound on the tail of the binomial distribution [17], we can derive the following fact regarding that range. The proof of this fact can be found in the extended version of this paper.

Assume that $\lambda = \frac{m}{\tau} \gg 1$, s.t. $\lambda + 1 \approx \lambda$. Define the entropy function of y, $0 \le y \le 1$, as $H(y) = -y \ln y - (1-y)\ln(1-y)$ and $H'(y) = dH(y)/dy$. For all $x \ge \lambda + 1$,

$$\frac{1}{2\sqrt{x\alpha(1-\alpha)}} e^{-xE(\alpha,\theta)} \le \sum_{j=\lambda+1}^{x} \binom{x}{j} \theta^j (1-\theta)^{x-j},$$

where $\alpha = \frac{\lambda+1}{x}$, $\theta = \frac{\tau}{\omega}$, and $E(\alpha,\theta) = H(\theta) + (\alpha - \theta)H'(\theta) - H(\alpha)$. Furthermore, if

$$x < \frac{m\omega}{\tau^2}, \tag{11}$$

then

$$\sum_{j=\lambda+1}^{x} \binom{x}{j} \theta^j (1-\theta)^{x-j} \le e^{-xE(\alpha,\theta)}.$$

According to [17], $E(\alpha,\theta) < 0$ when $x > \frac{m\omega}{\tau^2}$. So, when $x > \frac{m\omega}{\tau^2}$, the lower bound indicates that the tail of the binomial distribution increases exponentially with respect to x. It is also true that $E(\alpha,\theta) > 0$ when Inequality (11) is satisfied [17]. The upper bound indicates that the tail of the binomial distribution can be exponentially bounded away from 1 when x is not close to $\frac{m\omega}{\tau^2}$. For example, assume that x is 25% away from $\frac{m\omega}{\tau^2}$, i.e., $x = 0.75 * \frac{m\omega}{\tau^2} = 413$, where $m = 200, \tau = 2$, and $\omega = 11$, the upper bound is $e^{-5.089} = 0.006$ which is two orders of magnitude smaller than 1. Hence, $\frac{m\omega}{\tau^2}$ can be used as an estimation (upper bound) of the value of x where the fraction of the compromised secure communication links increases exponentially with respect to x. So the adversary can obtain higher payoff when the number of nodes it compromises reaches within the neighborhood of $\frac{m\omega}{\tau^2}$. The results shown in Fig. 4 verify that this estimation is quite accurate.

Based on the above discussions, the number of nodes an adversary needs to compromise to gain a significant payoff is linearly related to the amount of the memory used when ω and τ are fixed. That is, if the probability of any two nodes sharing at least one space, p_{actual}, is fixed, increasing the memory space at each node linearly increases the degree of security. For fixed memory usage, the security is linearly related to $\frac{\omega}{\tau^2}$. Since ω and τ are related to p_{actual}, one should choose those values of ω and τ that satisfy the requirement on global connectivity and at the same time yield largest value of $\frac{\omega}{\tau^2}$. For example, by using Inequality (4), one may find all the pairs of (ω, τ) that satisfy the requirement of the global connectivity. Among all the pairs, the one with the largest value of $\frac{\omega}{\tau^2}$ gives the best security strength.

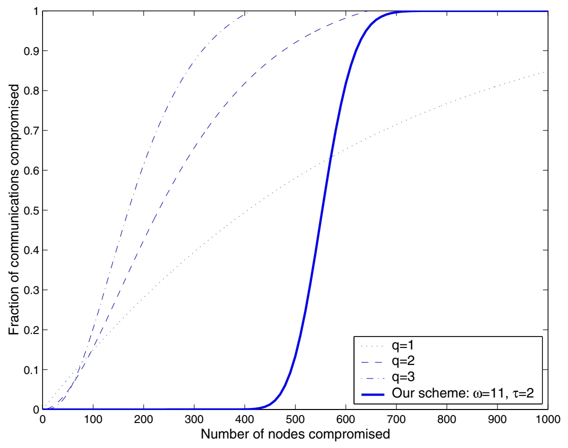
5. OVERHEAD ANALYSIS

5.1 Communication Overhead

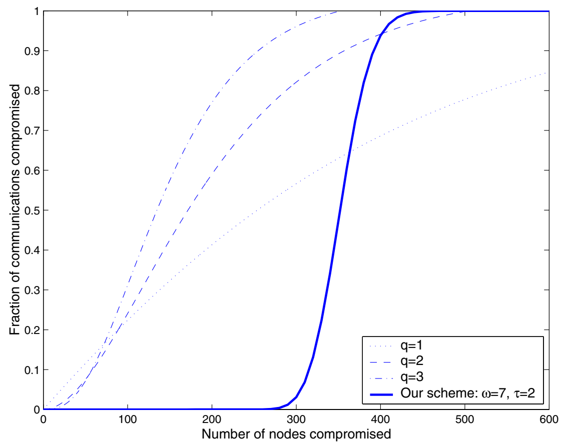
According to our previous discussions on p_{actual}, the probability that two neighbor nodes share a key space is less than 1. When two neighboring nodes are not connected directly, they need to find a route, in the key sharing sense, to connect to each other. We investigate the number of hops required on this route under various conditions for our scheme in this section. When the two neighbors are connected directly, the number of hops needed to connect them is obviously 1. When more hops are needed to connect two neighbor nodes, the communication overhead of setting up the security association between them is higher.

Let $p_h(\ell)$ be the probability that the smallest number of hops needed to connect two neighboring nodes is ℓ. Obviously, $p_h(1)$ is p_{actual}. We present the results of $p_h(2)$ and $p_h(3)$ as follows, while leaving the details of the calculation to the extended version of this paper:

$$\begin{aligned} p_h(2) \;=\; & (1 - p_{actual}) \\ & \cdot \left\{ 1 - 2\int_0^1 y p_{2,2}^{\frac{n}{\pi}\left[2cos^{-1}\left(\frac{y}{2}\right) - y \cdot \sqrt{1-\left(\frac{y}{2}\right)^2}\right]} dy \right\} \end{aligned}$$



(a) $m = 200, p_{actual} = 0.33$



(b) $m = 200, p_{actual} = 0.5$

Figure 4: The figures show the probability that a specific random communication link between two random nodes i, j can be decrypted by the adversary when the adversary has captured some set of x nodes that does not include i or j. m is the memory usage (m multiplied by the key length is the total amount of memory used for storing keys or key information), p_{actual} is the probability of any two neighbors being able to set up a secure link.

$$p_h(3) \approx [1 - p_h(1) - p_h(2)] \left[1 - 2\int_0^1 z \cdot (\tilde{p}_{3,2})^{\int_0^{2\pi}\int_0^1 \frac{n^2}{\pi^2}\left[2cos^{-1}(\frac{x}{2}) - x\sqrt{1-(\frac{x}{2})^2}\right]dy d\theta} dz\right]$$

where

$$p_{2,2} = 1 - \frac{\binom{\omega-\tau}{\tau}\left[\binom{\omega}{\tau} - 2\binom{\omega-\tau}{\tau} + \binom{\omega-2\tau}{\tau}\right]}{\binom{\omega}{\tau}^2}$$

$$\tilde{p}_{3,2} \approx 1 - \frac{\binom{\omega-\tau}{\tau}}{\binom{\omega}{\tau}^3} \cdot \sum_{a=1}^{\tau-1}\sum_{b=1}^{\tau-1}\sum_{c=1}^{\tau-\max(a,b)} \binom{\tau}{a}\binom{\tau}{b}\binom{\omega-2\tau}{c} \cdot \binom{\omega-2\tau-c}{\tau-a-c}\binom{\omega-2\tau-(\tau-a)}{\tau-b-c}$$

$$x = \sqrt{y^2 + z^2 + 2yz\cos(\theta)}.$$

We present the values of $p_h(1)$, $p_h(2)$, and $p_h(3)$ in Fig. 5. From these figures, we can observe that $p_h(1)$ and $p_h(2)$ add up to 1 when τ is large. So the communication overhead is limited to 2 hops when τ is large; when $n = 40$ and $p_{actual} > 0.3$, the overhead is bounded by 3 hops (recall that n is the expected number of neighbors within wireless communication range of a node).

5.2 Computational Overhead

As indicated in Section 2, it is necessary for nodes to calculate the common keys by using the corresponding columns of matrix G. If the Vandermonde matrix is chosen to be the G matrix, the dominating computation cost in our scheme is due to 2λ modular multiplications: $\lambda - 1$ come from the need to regenerate the corresponding column of G from a seed, the other $\lambda + 1$ come from the inner product of the corresponding row of $(DG)^T$ with this column of G. For example, to regenerate the first column of G, which consists of 1, s, s^2, …, s^λ, a node needs to compute s^2, …, s^λ; the total number of modular multiplications is $\lambda - 1$.

To analyze the computational overhead of these 2λ modular multiplications, we compare our computation with the RSA public key encryption algorithm, whose cost corresponding to modular multiplications makes it unsuitable for sensor networks. We want to show that the energy consumption of the modular multiplications in our scheme is far less than that of RSA. This is due to two factors: λ is small and the block size is small.

According to Equation (5), when $m = 200$ and $\tau = 4$, λ is about 50; the total number of multiplications is 100. If we choose 64 bits as the size of a secret key, then our modular multiplications are 64-bit computations. Therefore we need 100 64-bit modular multiplications. Compared to RSA, this is a very small number. In RSA signature signing scheme, the length for the exponent usually needs to be more than 1024 bits long, so the exponentiation requires at least 1024 multiplications. Moreover, using a 1024-bit exponent, RSA needs to be conducted in blocks that are at least 1024 bits long; a single modular multiplication on a 1024-bit block is $(\frac{1024}{64})^2 = 256$ times more expensive than a multiplication on a 64-bit block. Therefore, in total RSA scheme is about $256 * \frac{1024}{100} = 2621$ times more expensive than the multiplications in our scheme. Assuming that the energy cost is proportional to the cost of multiplications, the cost of our scheme is about $\frac{1}{2621}$ of the cost of RSA. According to the data presented by Carman, Kruus, and Matt [6], in a mid-range processor, such as the Motorola MC68328 "DragonBall", the cost of multiplications in our scheme is about 25 times more expensive than in an 128-bit AES encryption (AES is considered as very energy-efficient), i.e. the computation cost of our scheme is equivalent to encrypting a 3200-bit long message using AES.

Since the computation overhead occurs only once for each neighboring pair that has a common key space, the cost is not significant. Moreover, we can choose a larger τ to further lower the cost. However, our results show that increasing τ value may degrade the resilience of the network even though the connectivity is still the same. More analysis regarding this will be given in our future work.

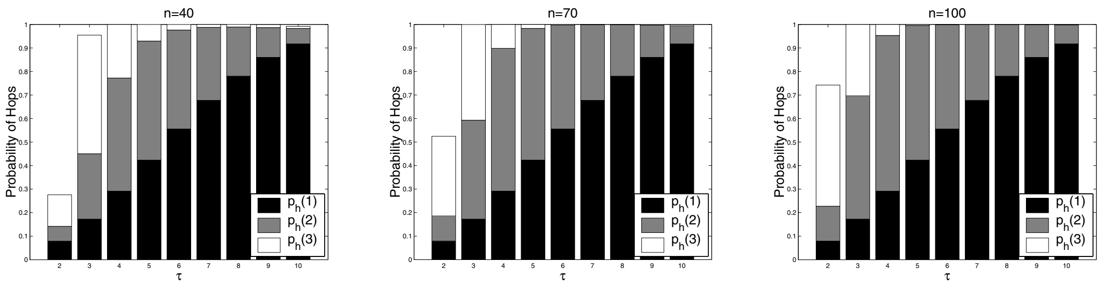


Figure 5: Communication Overhead Analysis ($\omega = 50$)

6. IMPROVING SECURITY USING TWO-HOP NEIGHBORS

In this section we describe a way to further improve the security of our key pre-distribution scheme. Based on Inequality (4), we have

$$1 - (1 - \frac{\tau}{\omega})(1 - \frac{\tau}{\omega - 1}) \cdots (1 - \frac{\tau}{\omega - \tau + 1})$$
$$\geq \frac{(N-1)}{nN}(\ln(N) - \ln(-\ln(P_c))). \quad (12)$$

Notice that the left side is smaller when ω is larger, and the right side is smaller when n is larger when other parameters are fixed. Therefore, when the network size N, the global connectivity P_c, and τ are fixed, we can select a larger ω if the expected number of neighbors n increases while still satisfying the above inequality. We know immediately from Inequality (11) that the larger the value of ω is, the more resilient the network will be. Therefore, increasing n can lead to security improvement.

There are two ways to increase n for an existing sensor network: the first is to increase the communication range, but this also increases energy consumption. The second way is to use two-hop neighbors. A two-hop neighbor of node v is a node that can be reached via one of v's one-hop (or direct) neighbors. To send a message to a two-hop neighbor, v needs to ask its direct neighbor to forward the message. Since the intermediate node only forwards the message and does not need to read the contents of the message, there is no need to establish a secure channel between the sender and the intermediate node, or between the intermediate node and the two-hop neighbor. As long as the sender and its two-hop neighbor can establish a secure channel, the communication between them will be secured.

If two nodes, i and j, are two-hop neighbors and both of them carry key information from a common key space, they can find a secret key between themselves using the following approach: First, they find an intermediate node I that is a neighbor to both of them. Nodes i and j then exchange their identities and public part of key space information via I. Then, i and j find a common key space, and compute their secret key in that common key space. i and j can then encrypt any future communication between themselves using this secret key. Although all future communication still needs to go through an intermediate node, e.g., I, the intermediate node cannot decrypt the message because it does not have the key.

After all direct neighbors and two-hop neighbors have established secure channels among themselves, the entire network forms an *Extended Key-Sharing Graph* G_{eks}, in which two nodes are connected by an edge if there is a secure channel between them, i.e. these two nodes (1) have at least one common key space, and (2) are either direct neighbors or two-hop neighbors. Once we have formed the G_{eks}, key agreement between any pair of two neighboring nodes i and j can be performed based on G_{eks} in the same way as it is performed based on the original Key-Sharing Graph G_{ks}. The difference between this scheme and the G_{ks}-based key agreement scheme is that in the G_{eks}-based key agreement scheme, some edges along a secure path might be an edge between two-hop neighbors, thus forwarding is needed.

6.1 Security Improvement

Security can be improved significantly if key agreement is based on G_{eks}. When we treat a two-hop neighbor as a neighbor, the radius of the range covered by a node doubles, so the area that a node can cover is increased by four times. Therefore, the expected number of neighbors n' for each node in G_{eks} is about four times as large as that in G_{ks}. According to Equations (1) and (2), to achieve the same connectivity P_c as that of G_{ks}, the value of $p_{required}$ for G_{eks} is one fourth of the value of $p_{required}$ for G_{ks}. Thus, the value of p_{actual} for G_{eks} is one fourth of the value of p_{actual} for G_{ks}. As we have already shown, when τ is fixed, the larger the value of ω is, the smaller the value of p_{actual} is. For example, assuming a network size $N = 10,000$ and the desirable connectivity $P_c = 0.99999$, if we fix $\tau = 2$, we need to select $\omega = 7$ for the G_{ks}-based key agreement scheme; however, using G_{eks}-based scheme, we can select $\omega = 31$. The security of the latter scheme is improved significantly. By using Equation (11), there is about $31/7 (\approx 4.5)$ times security improvement of the two-hop-neighbor scheme over the basic 1-hop-neighbor scheme. Using Equation (9), we plot the security property of the above two cases in Fig. 6.

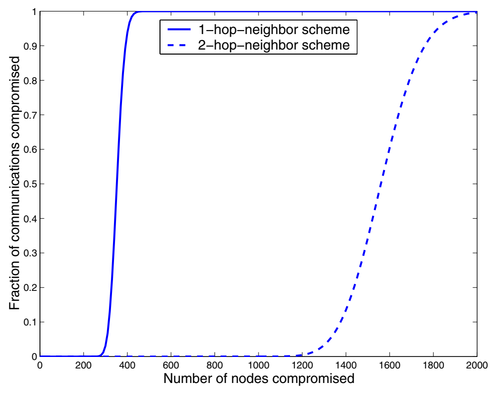


Figure 6: Comparison: The left curve uses the 1-hop-neighbor scheme (with $\omega = 7$ and $\tau = 2$), and the right curve uses the 2-hop-neighbor scheme (with $\omega = 31$, and $\tau = 2$). Both figures achieve the same desirable global connectivity $P_c = 0.99999$.

6.2 Overhead Analysis

Such security improvement does come with a cost. If the length (the total number of edges) of a path between two nodes in G_{eks} is ℓ, the actual number of hops along this path is larger than ℓ because some edges in G_{eks} connect two two-hop neighbors. For each node, the number of two-hop neighbors on the average is three times the number of one-hop neighbors if nodes are uniformly distributed. Therefore, assuming that the probability of selecting a two-hop edge and a one-hop edge is the same, for a path of length ℓ, the expected actual length is $\frac{3}{4} * 2\ell + \frac{1}{4} * \ell = 1.75\ell$ (note: in practice, we can achieve better than 1.75ℓ because we usually prefer the one-hop edge if both a one-hop edge and a two-hop edge are candidates for a secure path). Let $p'_h(\ell)$ be the $p_h(\ell)$ value of the two-hop-neighbor scheme and let $p''_h(\ell)$ be the $p_h(\ell)$ value of the basic scheme (only using direct neighbors); assume the maximum length of the shortest path between two neighbors is L. Therefore, the ratio between the overhead of the two-hop-neighbor scheme and that of the basic scheme can be estimated using the following formula:

$$\text{Relative Overhead} = \frac{p'_h(1) + \sum_{\ell=2}^{L} 1.75\ell \cdot p'_h(\ell)}{\sum_{\ell=1}^{L} \ell \cdot p''_h(\ell)}, \qquad (13)$$

where we do not need to multiply first term with 1.75 since if two neighbors share a common key, then the length of path between them is 1 and is never a two-hop edge. For example, the overhead ratio of the two schemes used in Fig. 6 is 3.18, namely with 3.18 times more overhead, the resilience can be improved by 4 times. The communication cost discussed here occurs only during the key setup phase, so it is a one-time cost. The idea of two-hop neighbors can be extended to multi-hop neighbors, and the security can be further improved.

7. CONCLUSIONS

We have presented a new pairwise key pre-distribution scheme for wireless sensor networks. Our scheme has a number of appealing properties. First, our scheme is scalable and flexible. For a network that uses 64-bit secret keys, our scheme allows up to $N = 2^{64}$ sensor nodes. These nodes do not need to be deployed at the same time; they can be added later, and still be able to establish secret keys with existing nodes. Second, compared to existing key pre-distribution schemes, our scheme is substantially more resilient against node capture. Our analysis and simulation results have shown, for example, that to compromise 10% of the secure links in the network secured using our scheme, an adversary has to compromise 5 times as many nodes as he/she has to compromise in a network secured by Chan-Perrig-Song scheme or Eschenauer-Gligor scheme. Furthermore, we have also shown that network resilience can be further improved if we use multi-hop neighbors.

We have conducted a thorough overhead analysis to show the efficiency of our scheme. The communication overhead analysis has shown that when $p_{actual} \geq 0.33$, a node can almost (with very high probability) reach its neighbor within at most 3 hops. For the computation overhead, although our scheme involves modular multiplications, we have shown that the energy cost is about the same as encrypting a message of length 3200 bits using AES.

8. REFERENCES

[1] Wireless Integrated Network Sensors, University of California, Available: http://www.janet.ucla.edu/WINS.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, August 2002.

[3] R. Anderson and M. Kuhn. Tamper resistance - a cautionary note. In *Proceedings of the Second Usenix Workshop on Electronic Commerce*, pages 1–11, November 1996.

[4] R. Blom. An optimal class of symmetric key generation systems. *Advances in Cryptology: Proceedings of EUROCRYPT 84 (Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, eds.), Lecture Notes in Computer Science, Springer-Verlag*, 209:335–338, 1985.

[5] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. *Lecture Notes in Computer Science*, 740:471–486, 1993.

[6] D. W. Carman, P. S. Kruus, and B. J. Matt. Constraints and approaches for distributed sensor network security. NAI Labs Technical Report #00-010, available at http://download.nai.com/products/media/nai/zip/nailabs-report-00-010-final.zip, 2000.

[7] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, pages 197–213, Berkeley, California, May 11-14 2003.

[8] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, November 1976.

[9] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. Technical Report, Syracuse University, July 2003. Available from http://www.cis.syr.edu/∼wedu/Research/paper/ddhcv03.pdf.

[10] Erdős and Rényi. On random graphs I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[11] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, November 2002.

[12] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: Mobile networking for smart dust. In *Proceedings of the 5th Annual ACM/IEEE Internation Conference on Mobile Computing and Networking (MobiCom)*, pages 483–492, 1999.

[13] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. New York, NY: Elsevier Science Publishing Company, Inc., 1977.

[14] D. Malkhi, M. Reiter, A. Wool, and R. N. Wright. Probabilistic quorum systems. *Information and Computation*, (2):184–206, November 2001.

[15] B. C. Neuman and T. Tso. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32(9):33–38, September 1994.

[16] A. Perrig, R. Szewczyk, V. Wen, D. Cullar, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *Proceedings of the 7th Annual ACM/IEEE Internation Conference on Mobile Computing and Networking (MobiCom)*, pages 189–199, Rome, Italy, July 2001.

[17] W. W. Peterson. *Error-Correcting Codes*. Cambridge, MA: Mass. Inst. Tech., second edition, 1972.

[18] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.