A Pairwise Key Pre-Distribution Scheme for Wireless Sensor Networks

WENLIANG DU and JING DENG
Syracuse University
YUNGHSIANG S. HAN
National Chi Nan University, Taiwan
PRAMOD K. VARSHNEY
Syracuse University
and
JONATHAN KATZ and ARAM KHALILI
University of Maryland, College Park

To achieve security in wireless sensor networks, it is important to be able to encrypt and authenticate messages sent between sensor nodes. Before doing so, keys for performing encryption and authentication must be agreed upon by the communicating parties. Due to resource constraints, however, achieving key agreement in wireless sensor networks is non-trivial. Many key agreement schemes used in general networks, such as Diffie-Hellman and other public-key based schemes, are not suitable for wireless sensor networks due to the limited computational abilities of the sensor nodes. Pre-distribution of secret keys for all pairs of nodes is not viable due to the large amount of memory this requires when the network size is large. To solve the key pre-distribution problem, two elegant key pre-distribution approaches have been proposed recently.

In this paper, we provide a framework in which to study the security of key pre-distribution schemes, propose a new key pre-distribution scheme which substantially improves the resilience of the network compared to previous schemes, and give an in-depth analysis of our scheme in terms of network resilience and associated overhead. Our scheme exhibits a nice threshold property: when the number of compromised nodes is less than the threshold, the probability that communications between any additional nodes are compromised 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 large fraction of the network before it can achieve any significant gain.

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

Additional Key Words and Phrases: Wireless sensor networks, key pre-distribution, security

This work was supported in part by grants ISS-0219560 and CCR-0310751 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. This paper is an extended version of [Du et al. 2003].

1. INTRODUCTION

Recent advances in electronic and computer technologies have paved the way for the proliferation of wireless sensor networks (WSNs). 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 [Kahn et al. 1999] and WINS.[1]

Sensor networks are being deployed for a wide variety of applications [Akyildiz et al. 2002], 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 these networks 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 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 schemes, public-key schemes, and key pre-distribution schemes. *Trusted-server* schemes depend on a trusted server for key agreement between nodes; an example is Kerberos [Neuman and Tso 1994]. This type of scheme is not suitable for sensor networks because in the locations where WSNs are deployed, one cannot generally assume that any trusted infrastructure is in place. *Public-key* schemes depend on asymmetric cryptography and require some sort of public-key infrastructure to be in place; an example of such schemes is an authenticated key agreement protocol using public-key certificates. However, as pointed out by Perrig, et al. [Perrig et al. 2001], the limited computation and energy resources of sensor nodes often make it undesirable to use public-key algorithms in WSNs. A third way to establish keys is via *pre-distribution*, where (secret) key information is distributed to all sensor nodes prior to deployment. Such schemes seem most appropriate for WSNs.

If it is known which nodes will be in the same neighborhood before deployment, pairwise keys can be established between these nodes (and only these nodes) *a priori*. However, most sensor network deployments are random; thus, such *a priori* knowledge about the topology of the network does not exist. A number of key pre-distribution schemes do not rely on prior knowledge of the network topology. A naive solution is to let all nodes store an identical *master* secret key. Any pair of nodes can use this master secret key to securely establish a new pairwise key. However, this scheme does not exhibit desirable network resilience: if a single node is compromised, the security of the entire sensor network is 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 [Anderson and Kuhn 1996].

At the other extreme, one might consider a key pre-distribution scheme in which each

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

sensor stores $N-1$ keys, each of which is known to only one other sensor node (here, we let N denote the total number of nodes in the network). This scheme guarantees perfect resilience because any number of compromised nodes does not affect the security of any *un*compromised pairs of nodes. Unfortunately, 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 when using this scheme because the existing nodes do not have the new nodes' keys.

Recently, two key pre-distribution schemes suited for sensor networks have been proposed. Eschenauer and Gligor [Eschenauer and Gligor 2002] proposed a random key pre-distribution scheme which may be summarized as follows: 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 a common key (if any) within their subsets and use that key as their shared secret key. Now, the existence of a shared key between a particular pair of nodes is not certain but is instead guaranteed only with some probability (which can be tuned by adjusting the parameters of the scheme). Eschenauer and Gligor note that this does not present an insurmountable problem as long as any two nodes can securely communicate via a sequence of secure links; see Sections 4 and 7 for further discussion.

Based on this scheme, Chan, Perrig, and Song [Chan et al. 2003] proposed a generalized "q-composite" scheme which improves the resilience of the network (for the same amount of key storage) and requires an attacker to compromise many more nodes in order to compromise any additional communication. The difference between this scheme and the previous scheme is that the q-composite scheme requires two nodes to find q (with $q > 1$) keys in common before deriving a shared key and establishing a secure communication link. It is shown that, by increasing the value of q, network resilience against node capture is improved for certain ranges of other parameters [Chan et al. 2003].

1.1 Main Contributions

The primary contribution of this work is a new key pre-distribution scheme which offers improved network resilience (for the same storage constraints) compared to the existing schemes mentioned above. The scheme requires more computation than previous schemes, but we show that this extra computation is small compared to that required by public-key schemes. We provide a thorough theoretical analysis of the security of our scheme, as well as its associated overhead. A high-level overview of this scheme, and a discussion of its advantages, appears below. As part of our analysis of the security of this scheme, we also introduce a rigorous *framework* (i.e., formal definitions of security) appropriate for analyzing key pre-distribution schemes for wireless sensor networks. Somewhat surprisingly, we found that prior definitions of security for key pre-distribution schemes were insufficient for our intended application; thus, we believe our framework is of independent interest and should prove useful for further work in this area.

Our key pre-distribution scheme extends and improves upon Blom's key pre-distribution scheme [Blom 1985] by combining this scheme with the random key pre-distribution methods discussed previously. Blom proposed a key pre-distribution scheme that allows *any* pair of nodes to find a secret pairwise key between them. Compared to the "trivial" scheme mentioned earlier in which each node stores $(N-1)$ keys, Blom's scheme only requires nodes to store $\lambda+1$ keys, where $\lambda \ll N$. 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 at most λ nodes,* *uncompromised nodes are perfectly secure. When an adversary compromises more than λ nodes, all pairwise keys in 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 required to store key information, as increasing λ leads to higher memory usage. The goal of our scheme is to increase the network's resilience against node capture in a probabilistic sense (and not in a perfect sense, as in the Blom scheme) without using too much memory.

Blom's scheme uses a *single* key space to ensure that any pair of nodes can compute a shared key. Motivated by the random key pre-distribution schemes presented previously [Eschenauer and Gligor 2002; Chan et al. 2003], we propose a new scheme using *multiple* key spaces. That is, 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. Now (by the properties of the underlying Blom scheme), if two nodes carry key information from a common space, they can compute a pairwise key. Of course, it is no longer certain that two nodes can generate a pairwise key (as in Blom's scheme); instead (as in previous random key pre-distribution schemes), we have only a probabilistic guarantee that this will be possible. Our analysis shows that using the same amount of memory (and for the same probability of deriving a shared key), our new scheme is substantially more resilient than previous probabilistic key pre-distribution schemes.

To further improve the resilience of our approach while maintaining connectivity of the network, we develop a two-hop-neighbor key pre-distribution scheme. The idea is to let the direct neighbor of a sender forward messages, so that nodes that are two hops away from the sender can also receive them. 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 by a factor of four compared to our initial scheme.

The remainder of this paper is organized as follows. Section 2 describes our proposed framework for analyzing the security of key pre-distribution schemes in terms of their effectiveness in establishing "secure channels" between the network nodes.[2] We also show a simple method to convert any secure key pre-distribution scheme into a scheme for establishing secure channels. Section 3 reviews Blom's key pre-distribution scheme which will be used as a building block for our main key pre-distribution scheme, described in Section 4. Section 5 rigorously quantifies the resilience of our scheme to node capture, and compares our scheme with existing key pre-distribution schemes. Section 6 presents the communication and computation overheads of our scheme. Section 7 describes our two-hop-neighbor key pre-distribution scheme. We conclude in Section 8.

1.2 Other Related Work

The Eschenauer-Gligor scheme [Eschenauer and Gligor 2002] and the Chan-Perrig-Song scheme [Chan et al. 2003] have been reviewed earlier in this section. Detailed comparisons with these two schemes will be given in Section 5.

Blundo et al. proposed several schemes allowing any group of n parties to compute

[2]Interestingly, secure key pre-distribution is necessary, but *not* sufficient, for establishing secure channels.

a common key which is perfectly secret with respect to any coalition of t other parties [Blundo et al. 1993]. When $n = 2$, their main scheme may be viewed as a special case of Blom's scheme [Blom 1985], which is reviewed in Section 3. Although both Blom's scheme (for $n = 2$) and the main scheme of Blundo, et al. (for arbitrary n) match the known lower bound [Blundo et al. 1993] in terms of their memory usage for any desired resilience t, we stress that this lower bound holds *only* when (1) *all* groups of size n must be able to compute a shared key and (2) the network must be *perfectly* resilient to at most t captured nodes. By relaxing these requirements (slightly) and considering the *probabilistic* analogues of the above, we obtain more memory-efficient schemes.

Perrig et al. proposed SPINS [Perrig et al. 2001], a security architecture in which each sensor node shares a secret key with a base station. In this scheme, two sensor nodes cannot directly establish a secret key; however, they can set up a shared key using the base station as a trusted third party. The scheme described in this work does not rely on any trusted parties after nodes have been deployed.

A similar approach to the one described in this paper was independently developed by Liu and Ning [Liu and Ning 2003], which was published at the same time as the conference version of this paper [Du et al. 2003]. Compared to [Liu and Ning 2003], this paper provides a more thorough theoretical security analysis and communication overhead analysis; we also introduce a rigorous *framework* (i.e., formal definitions of security) appropriate for analyzing key pre-distribution schemes for wireless sensor networks. Moreover, we also describe a further improvement using multi-hop neighbors.

2. A SECURITY FRAMEWORK FOR KEY PRE-DISTRIBUTION SCHEMES

Before describing our primary scheme in detail, we first propose a general framework in which to analyze the security of key pre-distribution schemes in general. Our starting point is the following simple observation (which, however, we found lacking in previous work): the goal of a key pre-distribution scheme is not simply to distribute keys, but rather to distribute keys *which can then be used to secure network communication*. While the former is necessary for the latter, it is decidedly *not* sufficient. As an example, we show below that although the Eschenauer-Gligor scheme ensures that the key K_{ij} established by some pair of nodes i and j remains unknown to an adversary (with high probability, for some fraction of compromised nodes), the scheme is *insecure* if K_{ij} is used to authenticate the communication between these nodes. Related problems arise in the schemes of Blom, Blundo, et al., and Chan-Perrig-Song, as well. This observation emphasizes the importance of precise definitions of security, as well as rigorous proofs in some well-defined model.

We develop the framework as follows: We first define key pre-distribution schemes, and describe for such schemes a "basic" level of security. This definition (informally) captures the idea that an adversary should be unable to determine the key shared by some pair of users (except with low probability), and roughly corresponds to the level of security considered in previous work in this area. We then define a stronger notion of security which we believe more accurately represents the level of security expected from key pre-distribution schemes when used *in practice*. We focus specifically on the case of message authentication, yet our results easily extend to the case of pairwise encryption. Our definition (informally) requires that an adversary be unable to insert a bogus message which is accepted as legitimate by one of the nodes (except with low probability). Schemes meeting this, more stringent notion of security are said to achieve *secure pairwise authentication*.

After introducing these definitions, we show that a scheme meeting the "basic" notion of security does not necessarily achieve secure pairwise authentication. On a more positive note, we show a simple way to convert any scheme achieving the "basic" level of security to one which *does* achieve secure pairwise authentication.

Our definitions, as well as our results, are described here in a relatively informal fashion. Yet, it should be straightforward for the interested reader to derive formal definitions and statements of our results from the discussion below.

We begin with a discussion of key pre-distribution schemes. We view such schemes as being composed of algorithms for key generation, key distribution, and key derivation. In the randomized *key generation* phase, some master secret information S is established. Given S and a node identity i, a deterministic *key distribution* algorithm generates information k^i which will be stored by node i. Finally, during the *key derivation* phase, two distinct nodes i and j holding k^i and k^j, respectively, execute an algorithm Derive and output a shared key $K_{ij} \in \{0,1\}^\ell$ or $\perp$ if no such key can be established. We denote execution of this algorithm by node i (holding information k^i) as $\mathsf{Derive}(k^i, i, j)$; we always require $\mathsf{Derive}(k^i, i, j) = \mathsf{Derive}(k^j, j, i)$. We assume the key derivation stage is deterministic, but allow that it may require interaction between nodes i and j. Note that a pair of nodes i, j is not guaranteed to be able to establish a shared key $K_{ij} \neq \perp$. We assume that the probability (over choice of master key S) that i and j can establish a shared key is the same for any $i \neq j$, and refer to this as the *connectivity* (denoted by p) of the scheme.

We define our "basic" level of security via the following game: Fix node identities i and j, and run an instance of the key pre-distribution scheme. An adversary is given the information $\{k^{i_1}, \ldots, k^{i_t}\}$ for t randomly-selected nodes, where neither i nor j are in the set $\{i_1, \ldots, i_t\}$ (this models adversarial compromise of these nodes, with concomitant exposure of the secret information stored thereon). The adversary "succeeds" if: (1) $K_{ij} \neq \perp$, and (2) the adversary can successfully output the key K_{ij}. We will say that a key pre-distribution scheme is (t, ϵ)-secure if, for any i, j, the probability that an adversary succeeds is at most ϵ. (In the above formulation, we have not restricted the computational abilities of the adversary in any way. Clearly, this relaxation can also be considered.) Note that meaningful security is obtained only when $\epsilon < p$, since the first condition (i.e., $K_{ij} \neq \perp$) only holds with probability p.

Before introducing a more useful notion of security, we define a pairwise authentication scheme. This is simply a key pre-distribution scheme with an additional *message authentication* algorithm Mac and *message verification* algorithm Vrfy. Now, if nodes i, j establish a shared key $K_{ij} \neq \perp$, node i can authenticate its communication to node j as follows (j can authenticate its communication to i similarly): before sending message m, node i computes $\mathsf{tag} = \mathsf{Mac}_{K_{ij}}(m)$ and sends tag along with m; upon receiving (m, tag), node j accepts m only if $\mathsf{Vrfy}_{K_{ij}}(m, \mathsf{tag}) = 1$. For completeness, we define $\mathsf{Mac}_\perp(m) = \perp$ for all m, and $\mathsf{Vrfy}_\perp(m, \mathsf{tag}) = 0$ for all m, tag.

We now define secure pairwise authentication via the following game: Fix i and j, and run an instance of the pairwise authentication scheme. An adversary is given $\{k^{i_1}, \ldots, k^{i_t}\}$ as before. Additionally, the adversary can repeatedly make an unlimited number of message authentication requests of the form $\overline{\mathsf{Mac}}(i', j', m)$, with the effect that node i' authenticates message m for node j' (using key K_{ij}) and returns the resulting tag to the adversary. (We stress that $i', j' \in \{i, j\}$ is allowed). Finally, the adversary outputs (m^*, tag^*) and "succeeds" if: (1) $\mathsf{Vrfy}_{K_{ij}}(m^*, \mathsf{tag}^*) = 1$ (in particular, this will require $K_{ij} \neq \perp$), and

(2) the adversary had never requested $\overline{\mathsf{Mac}}(i, j, m^*)$ or $\overline{\mathsf{Mac}}(j, i, m^*)$. Success corresponds to the adversary "inserting" the bogus message m^* which is accepted as valid by one of i, j even though neither node authenticated this message. Finally, we say that a scheme is a (t, ϵ)-secure *pairwise authentication scheme* if, for any (i, j), the probability that a polynomial-time adversary succeeds in the above game is at most ϵ. Note that we must now limit the computational abilities of the adversary since secure message authentication for an unbounded number of messages is impossible otherwise.

It is instructive to note that a key pre-distribution scheme secure in the basic sense need *not* be a secure pairwise authentication scheme. For example, consider a scheme in which K_{ij} is equal to $K_{i'j'}$ (for some $(i', j') \neq (i, j)$) with some high (i.e., non-negligible) probability; this is true for both the Eschenauer-Gligor and Chan-Perrig-Song schemes. Now, even if an adversary does not compromise *any* nodes, and even if it cannot guess K_{ij} (and hence the scheme remains secure in the basic sense), the scheme is not a secure pairwise authentication scheme. In particular, an adversary can take messages that were authenticated by i' and intended for j', and send these messages to j while claiming they originated from i; with high probability (namely, whenever $K_{i'j'} = K_{ij}$), the adversary's insertion goes undetected.

This problem of "repeated keys" has been noticed (although informally) in previous work. However, we stress that subtle problems may arise even when the probability of "repeated keys" is small. Whenever the keys used by different pairs of parties are not *independent* (in a probabilistic sense), a formal proof of secure pairwise authentication will not be possible in general. In fact, this reflects a serious potential vulnerability, as the presence of dependent keys leaves open the possibility of *related-key attacks* on the message authentication code or the lower-level primitives (i.e., block ciphers) from which the MAC is constructed. The possibility of such related-key attacks also rules out the easy "fix" in which nodes pre-pend the identities of the sender/receiver to any authenticated messages; this does nothing to prevent related-key attacks.

Given the above, one should focus on designing secure pairwise authentication schemes rather than secure key pre-distribution schemes. Luckily, it is simple to derive the former from the latter as follows: Let K_{ij} be the key derived by nodes i and j in some key pre-distribution scheme which is assumed to be secure in the basic sense discussed above. These nodes then compute $K'_{ij} = H(i, j, K_{ij})$, where H is a hash function modeled as a *random oracle* [Bellare and Rogaway 1993]. This key K'_{ij} is then used by i and j (as the key for *any* secure MAC) to authenticate their communication as suggested above. It can be shown that if the initial scheme is (t, ϵ)-secure in the basic sense, and if the probability of forgery for the MAC is ϵ', then the modified scheme is a $(t, q\epsilon + \epsilon')$-secure pairwise authentication scheme, where q is a bound on the number of hash function queries made by an adversary. The proof is straightforward, and is omitted here.

Since one may always convert any secure key pre-distribution scheme into a secure pairwise authentication scheme, we will analyze the security of our proposed scheme in the "basic" sense with the understanding that the above transformation should be applied before the scheme is used in practice. This modular analysis of security is (we believe) simpler, more intuitive, and less prone to error.

3. 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 derive a pairwise secret key [Blom 1985]. As long as no more than λ nodes are compromised, the network is perfectly secure (we call this the λ-secure property). We briefly describe how Blom's λ-secure key pre-distribution system works (we have made some slight modifications to the scheme in order to make it more suitable for sensor networks, for the essential features remain unchanged).

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 and $q > N$. Matrix G is 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.[3]

Therefore, when nodes i and j need to establish pairwise key, 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 shown [Blom 1985] that the above scheme is λ-secure if any $\lambda + 1$ columns of G are linearly independent. This λ-secure property guarantees that no coalition of up to λ nodes other than i and j can compute K_{ij} or K_{ji}.

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 $GF(q)$, we must set $|q|$ to be larger than the key size we desire. Thus, if 64-bit keys are desired we may choose q as the smallest prime number larger than 2^{64} (alternately, we may choose $q = 2^{64}$); note that for all reasonable values of N we will have $q > N$ as required. Let s be a primitive element of $GF(q)$; that is, each nonzero element in $GF(q)$ can be represented by some power of s. A feasible G can be

[3] We will show later that a sensor need not store the whole column, because each column can be generated from a single field element.

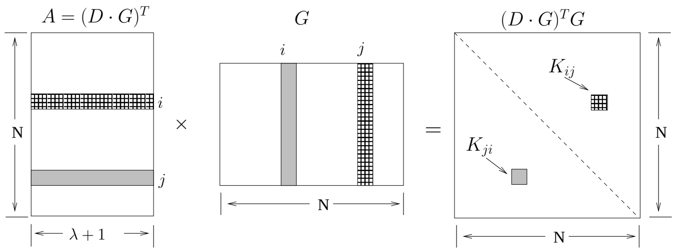


Fig. 1. Generating Keys in Blom's Scheme

designed as follows [MacWilliams and Sloane 1977]:

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

It is well-known that $s^i \neq s^j$ if $i \neq j \bmod q$ (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 [MacWilliams and Sloane 1977]. 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. Tradeoffs between memory usage and computational complexity will be discussed later in the paper.

4. 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 establish a shared secret key. If we imagine a graph in which each sensor node is a vertex and there is an edge between nodes only if they can establish a shared key, then Blom's scheme results in a *complete* graph (i.e., and 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. Previous work shows that *by requiring the graph to be connected rather than complete, the information stored by each sensor node can be reduced*.

Before we describe our proposed scheme, we define a *key space* (or *space* in short) as a pair of matrices (D, G) 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 pairwise key if they have picked a common key space.

4.1 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 establish a shared, secret key. Assume that each sensor node has a unique identity, whose range is from 1 to N. We also select the parameters τ, ω, and λ, where $2 \le \tau < \omega$. These parameters determine the security and performance of our scheme, as will be discussed later in the paper. Our key generation/distribution phase consists of the following steps:

Step 1: Generating G matrix. We first select a primitive element from a finite field $GF(q)$, where $|q|$ is larger than the desired key length (and also $q > N$), 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 3, although $G(j)$ contains $(\lambda + 1)$ elements, each sensor only needs to store one field element (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 as a node identity.

Step 2: Generating D matrices. We generate ω random, symmetric matrices $D_1, \ldots, D_\omega$ of size $(\lambda+1) \times (\lambda+1)$. We call each tuple $S_i = (D_i, G)$ (for $i = 1, \ldots, \omega$), 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; under no circumstance should a node send this information to any other node. According to Blom's scheme, two nodes can establish 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 (roughly) the same as the length of the secret keys, the memory usage of each node is $(\lambda + 1)\tau$ times the length of the key (we do not count the space required to store the seed for $G(j)$, since this may serve as the node identity).

4.2 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,[4] and (3) the seed of the column of G it carries.[5]

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, say 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 4.1. *(Key-sharing graph)* Let V represent all the nodes in the sensor net-

[4] 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 [Chan et al. 2003].

[5] As mentioned earlier, we could also let the node identity be the same as the seed.

work. 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 (i.e., are within wireless transmission range).

We now show how two neighboring nodes, i and j who do not share a common key space could still establish a shared secret key. 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, ..., v_t, j. To establish a common secret key between i and j, node i first generates a random key K. Then i sends the key to v_1 using their secure link; 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 determine the key.

4.3 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 such that the graph G_{ks} is connected with high probability? We use the following three-step approach, which is adapted from [Eschenauer and Gligor 2002]. Although this approach is heuristic and not rigorous, it has been suggested and used in previous work in this area [Eschenauer and Gligor 2002; Chan et al. 2003].

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., the probability they can establish a common key). 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*, and denote it by $p_{required}$.

Using results from the theory of random graphs [Erdős and Rényi 1959], we can relate the average node degree d (i.e., the average number of edges connected to each node) to the global connectivity probability P_c for a network of size N (for N large):

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

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 should be at least d as calculated above, the required local connectivity $p_{required}$ can be estimated as:

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

We stress that this only guarantees connectivity in a heuristic (and not a rigorous) sense: to apply the theory of random graphs it must be the case that a node has edges *with other nodes uniformly distributed throughout the graph.* Here, however, nodes only have edges to their physically-close neighbors. Yet, we are not aware of any problems in practice with using this heuristic estimate.

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., the probability that they can establish a common key). 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 value of ω must be at most 25 in order to achieve local connectivity $p_{actual} \geq 0.5$.

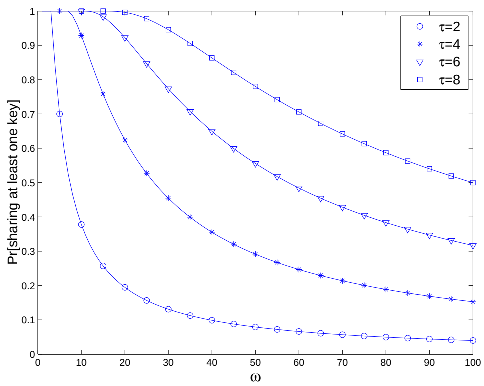


Fig. 2. Probability of sharing at least one key when two nodes each randomly chooses τ spaces from ω spaces.

The collection of sets of spaces assigned to each sensor form a probabilistic quorum system [Malkhi et al. 2001]: the desire is that every two sensors have a space in common with high probability. Next we show that if $\tau \geq \sqrt{\ln \frac{1}{1-p_{actual}}}\sqrt{\omega}$, then the probability of intersection is at least p_{actual}. For example, when $\tau \geq \sqrt{\ln 2}\sqrt{\omega}$, the probability of intersection is at least $1/2$. This helps explain the behavior evidence in Fig. 2. A proof of this fact, similar to proof of the "birthday paradox", is as follows: It is well-known that $1 - x \leq e^{-x}$ for all $x \geq 0$. Therefore,

$$\begin{aligned} p_{actual} &= 1 - \frac{((\omega-\tau)!)^2}{(\omega-2\tau)!\omega!} \\ &= 1 - \left(1 - \frac{\tau}{\omega}\right)\left(1 - \frac{\tau}{\omega-1}\right)\cdots\left(1 - \frac{\tau}{\omega-\tau+1}\right) \\ &\geq 1 - e^{-\left(\frac{\tau}{\omega}+\frac{\tau}{\omega-\tau}+\cdots+\frac{\tau}{\omega-\tau+1}\right)} \\ &\geq 1 - e^{-\frac{\tau^2}{\omega}}. \end{aligned}$$

Accordingly, we have

$$\tau \leq \sqrt{\ln \frac{1}{1 - p_{actual}}}\sqrt{\omega}.$$

Thus, the value of τ to achieve p_{actual} (for given ω) is at most $\sqrt{\ln \frac{1}{1-p_{actual}}}\sqrt{\omega}$.

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 $p_{actual} \geq p_{required}$. Thus:

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

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:

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

(As mentioned earlier, we do not count the field element needed to generate $G(i)$ since this can also serve as the node identity.)

5. 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? 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 the expected payoff an adversary obtains after capturing a certain number of nodes.

5.1 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). In Blom's scheme, for a space to be λ-secure, each node needs to use memory of size $\lambda+1$. Therefore, if the memory usage is m and each node needs to carry τ spaces, the value of λ should be $\lfloor \frac{m}{\tau} \rfloor - 1$. We use this value for λ in the following analysis.

Let $\mathcal{S}_i$ be the event that space S_i is broken (for $i = 1, \ldots, \omega$), and let $\mathcal{C}_x$ 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 let $\theta = \frac{\tau}{\omega}$. 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 be broken only after at least $\lambda + 1$ nodes are compromised (by the λ-secure property of the underlying Blom scheme), 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:

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

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, that 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 reasonably-high 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 directly for authentication (note, however, that this will *not* guarantee secure pairwise authentication in the sense of Section 2). After a space is broken, however, an adversary can generate all the pairwise keys in that space, and keys in that space can no longer be used for authentication purposes.

5.2 The Fraction of Network Communication 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

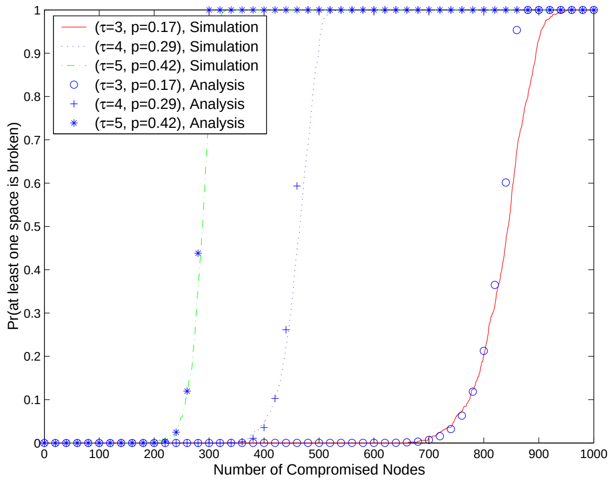


Fig. 3. The probability of at least one key space being compromised by the adversary when the adversary has captured x nodes ($m = 200$, $\omega = 50$). p in the figure represents p_{actual}.

want to find out the fraction of additional communication (i.e., communication among the 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 uncompromised nodes, 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),

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

$\Pr(S_1$ 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,

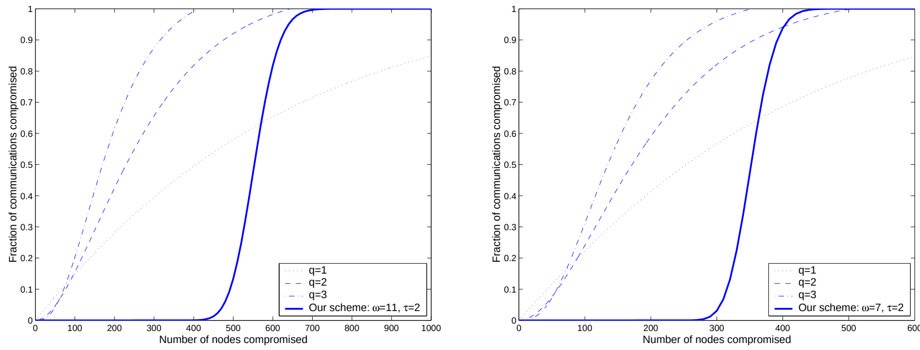
$$\begin{aligned}&\Pr(c \text{ is broken} \mid \mathcal{C}_x) \\ =\ &\omega \Pr(\mathcal{B}_1 \mid \mathcal{C}_x) = \omega \cdot \frac{1}{\omega} \cdot \Pr(S_1 \text{ is compromised} \mid \mathcal{C}_x) \\ =\ &\Pr(S_1 \text{ is compromised} \mid \mathcal{C}_x) \\ =\ &\sum_{j=\lambda+1}^{x} \binom{x}{j}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

Assume that there are γ secure communication links that do not involve any of the x compromised nodes. Given the probability $\Pr(c$ 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 follows directly from the linearity of expectations.

5.2.1 *Comparison.* Fig. 4 compares 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 advantages of our scheme. For example, in both the Chan-Perrig-Song and Eschenauer-Gligor schemes, when $m = 200$ and $p_{actual} = 0.33$ an adversary needs to compromise less than 100 nodes in order to compromise 10% of the links. In our scheme, however, the adversary needs to compromise 500 nodes before compromising 10% of the links. Therefore, our scheme quite substantially lowers the initial payoff to the adversary for smaller-scale network breaches. Chan, Perrig, and Song also proposed a modification of their scheme using multipath key reinforcement to improve the security [Chan et al. 2003]. The same technique can be applied to our scheme to improve the security as well; we leave further comparison to our future work.



(a) $m = 200$, $p_{actual} = 0.33$ (b) $m = 200$, $p_{actual} = 0.5$

Fig. 4. The figures show the probability that a specific random communication link between two random nodes i, j is compromised after an adversary has captured x nodes, not including i or j. The variable m denotes the memory usage (where the unit of memory is the length of a shared key), and p_{actual} denotes the probability that a random pair of nodes can establish a secure link.

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

5.2.2 *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 [Peterson 1972], we can derive the following theorem regarding that range.

THEOREM 5.1. *Assume that $\lambda = \frac{m}{\tau} \gg 1$, s.t. $\lambda + 1 \approx \lambda$. Define the entropy function of y, $0 \leq y \leq 1$, as $H(y) = -y \ln y - (1-y)\ln(1-y)$ and $H'(y) = dH(y)/dy$. For all $x \geq \lambda + 1$,*

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

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

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

PROOF. Assume that $x \geq \lambda + 1$. According to the bound on the tail of binomial distribution [Peterson 1972], Equation (9) can be bounded as follows:

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

and if $\alpha > \theta$, then

$$\sum_{j=\lambda+1}^{x}\binom{x}{j}\theta^j(1-\theta)^{x-j} \;\leq\; \alpha^{-\alpha x}(1-\alpha)^{-(1-\alpha)x}\theta^{\alpha x}(1-\theta)^{(1-\alpha)x}, \tag{12}$$

where $\alpha = \frac{\lambda+1}{x}$ and $\theta = \frac{\tau}{\omega}$. Since $\lambda = \frac{m}{\tau} \gg 1$, $\lambda + 1 \approx \lambda$. Consequently, $\alpha \approx \frac{\lambda}{x} = \frac{m}{\tau x}$. By taking the logarithm of the upper bound of Inequality (12) and multiplying by $-\frac{1}{x}$, we have

$$\begin{aligned} & -\frac{1}{x}\ln\left(\alpha^{-\alpha x}(1-\alpha)^{-(1-\alpha)x}\theta^{\alpha x}(1-\theta)^{(1-\alpha)x}\right) \\ = \; & -H(\alpha) - \alpha\ln\theta - (1-\alpha)\ln(1-\theta) \\ = \; & -H(\alpha) + H(\theta) + (\theta-\alpha)\ln\theta + [(1-\theta)-(1-\alpha)]\ln(1-\theta) \\ = \; & -H(\alpha) + H(\theta) + (\alpha-\theta)(-\ln\theta + \ln(1-\theta)). \end{aligned}$$

Since $H'(y) = dH(y)/dy = \ln(1-y) - \ln y$,

$$-\frac{1}{x}\ln\left(\alpha^{-\alpha x}(1-\alpha)^{-(1-\alpha)x}\theta^{\alpha x}(1-\theta)^{(1-\alpha)x}\right) = E(\alpha,\theta)$$

where

$$E(\alpha,\theta) = H(\theta) + (\alpha-\theta)H'(\theta) - H(\alpha).$$

Finally,

$$\begin{aligned} \alpha > \theta \iff & \frac{m}{x\tau} > \frac{\tau}{\omega} \\ \iff & x < \frac{m\omega}{\tau^2}, \end{aligned} \tag{13}$$

giving the claimed result. □

According to [Peterson 1972], $E(\alpha,\theta) < 0$ 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 [Peterson 1972]. 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, when x is 25% away from $\frac{m\omega}{\tau^2}$ (i.e., $x = 0.75 * \frac{m\omega}{\tau^2} = 413$) and $m = 200, \tau = 2$, and $\omega = 11$, then 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) on the value of x where the fraction of compromised links increases exponentially with respect to x. So the adversary can obtain higher payoff when the number of nodes it compromises is close to $\frac{m\omega}{\tau^2}$. The results shown in Fig. 4 verify that this estimation is quite accurate.

Based on the above discussion, 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, then 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 the largest value of $\frac{\omega}{\tau^2}$. For example, by using Inequality (4), one may find all pairs (ω, τ) satisfying the requirement on the global connectivity. Among all the pairs, the one with the largest value of $\frac{\omega}{\tau^2}$ gives the best security.

6. OVERHEAD ANALYSIS

6.1 Communication Overhead

According to our previous discussions, the probability p_{actual} 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 path (in the key-sharing graph) to connect to each other. In this section, we investigate the number of hops required on this path for various parameters of our scheme. 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}. For $p_h(2)$, the third node connecting these two nodes must be in the overlapped region of the transmission range of node i and node j, as shown in Fig. 5.

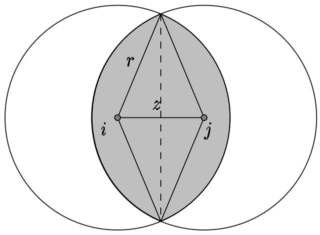


Fig. 5. Overlap Region $A_{overlap}(z)$

The size of this overlapped region is:

$$A_{overlap}(z) = 2r^2 \cos^{-1}\left(\frac{z}{2r}\right) - z \cdot \sqrt{r^2 - \left(\frac{z}{2}\right)^2}, \tag{14}$$

where r is the transmission range of each node. The total number of nodes in the overlap region is:

$$N_{overlap}(z) = \frac{n}{\pi r^2} A_{overlap}(z),$$

where n is the total number of sensor nodes in the transmission range of a sensor node.

We then calculate $p_h(2, z)$, the probability that i and j are not connected directly but there exists at least a common neighbor connecting them, given that the distance between i and j is z:

$$p_h(2, z) = (1 - p_{actual})[1 - p_{2,1}(z)]$$

where $p_{2,1}(z)$ is the probability that none of the common neighbors of i and j is connected to both of them given that i and j are not connected.

The value of $p_h(2)$ can be calculated as the average of $p_h(2, z)$ throughout all the possible values of z:

$$p_h(2) \;=\; \int_0^r f(z)p(2, z)dz$$

where $f(z)$ is the Probability Density Function (PDF) of z:

$$f(z) = \frac{\partial F(Z)}{\partial z} = \frac{\partial\,[Pr(Z \leq z)]}{\partial z} = \frac{\partial}{\partial z}\left[\frac{\pi z^2}{\pi r^2}\right] = \frac{2z}{r^2}.$$

A similar approach may be used to calculate $p_h(3)$. The only difference is that, in the case of $p_h(3)$, we need to find the probability that two nodes, nodes u and v, that are neighboring to nodes i and j, respectively, should provide a secure link between nodes i and j, as shown in Fig. 6.

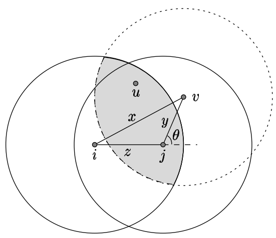


Fig. 6. Overlap Region for $p_h(3)$

We provide the full derivations of $p_h(2)$ and $p_h(3)$ in Appendix A. The final results are as follows:

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

$$\begin{aligned} p_h(3) \;&\approx\; [1 - p_h(1) - p_h(2)] \left[1 - 2\int_0^1 z \right. \\ &\left. \cdot\, (\tilde{p}_{3,2})^{\int_0^{2\pi}\int_0^1 \frac{n^2}{\pi^2}\left[2cos^{-1}\left(\frac{x}{2}\right) - x\sqrt{1-\left(\frac{x}{2}\right)^2}\right] dyd\theta} dz \right] \end{aligned}$$

where

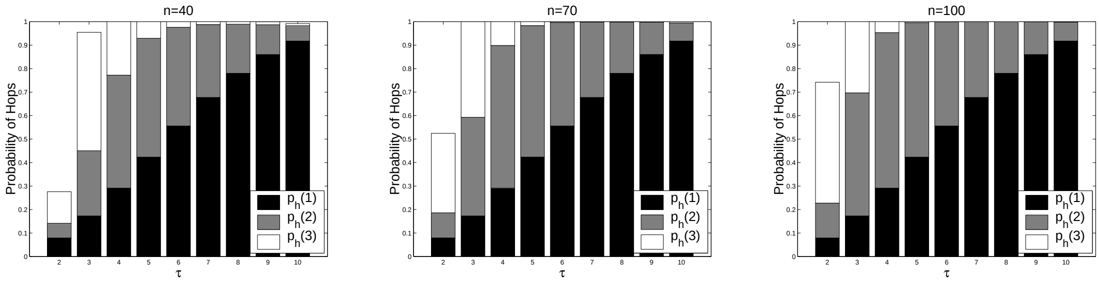


Fig. 7. Distribution on the number of hops required to connect neighbors ($\omega = 50$)

.

$$
\begin{aligned}
p_{2,2} &= 1 - \frac{\binom{\omega-\tau}{\tau}\left[\binom{\omega}{\tau} - 2\binom{\omega-\tau}{\tau} + \binom{\omega-2\tau}{\tau}\right]}{\binom{\omega}{\tau}^2} \\
\tilde{p}_{3,2} &\approx 1 - \frac{\binom{\omega-\tau}{\tau}}{\binom{\omega}{\tau}^3} \cdot \sum_{a=1}^{\tau-1}\sum_{b=1}^{\tau-1}\sum_{c=1}^{\tau-\max(a,b)} \binom{\tau}{a}\binom{\tau}{b}\binom{\omega-2\tau}{c} \\
&\quad \cdot \binom{\omega-2\tau-c}{\tau-a-c}\binom{\omega-2\tau-(\tau-a)}{\tau-b-c} \\
x &= \sqrt{y^2+z^2+2yz\cos(\theta)}.
\end{aligned}
$$

We plot the values of $p_h(1)$, $p_h(2)$, and $p_h(3)$ in Fig. 7. From these figures, we can observe that $p_h(1) + p_h(2) \approx 1$ when τ is large (i.e., the probability that at most 2 hops are required is essentially 1).

6.2 Computational Overhead

As indicated in Section 3, it is necessary for nodes to calculate the common keys by using the corresponding columns of matrix G. If the Vandermonde matrix is chosen as the G matrix, the dominating computation cost in our scheme is due to $2\lambda - 1$ multiplications in the field $GF(q)$: $\lambda - 1$ come from the need to regenerate the corresponding column of G from a seed, while the other λ come from the inner product of the corresponding row of $(DG)^T$ with this column of G. Note that this can easily be reduced to only λ multiplications using Horner's rule for polynomial evaluation. (Although $O(\lambda)$ additions in $GF(q)$ are also necessary, these are dominated by the field multiplications.)

To analyze the computational overhead of these modular multiplications, we compare our computation with the RSA public key encryption algorithm, whose cost makes it unsuitable for sensor networks. We 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 then roughly 100 (this assumes a naive implementation which does not apply Horner's rule). If we choose 64 bits as the size of a secret key, then our modular multiplications are 64-bit computations. In total, then, we need roughly 100 64-bit modular multiplications. For the RSA signature scheme using a 1024-bit modulus, the length of the private exponent usually needs to be roughly 1024 bits as well. Thus, a single exponentiation requires approximately 1500 multiplications. Moreover, a single multiplication modulo a 1024-bit integer is roughly $(\frac{1024}{64})^2 = 256$ times more expensive than a single multiplication modulo a 64-bit number. Therefore, computing an RSA signature is roughly $256 * \frac{1500}{100} = 3840$ times more expensive than deriving a shared key in our scheme. Assuming that the energy cost is proportional to the number of multiplications, the cost of our scheme is about $\frac{1}{3840}$ that of RSA. In a mid-range processor such as the Motorola MC68328 "DragonBall" (see [Carman et al. 2000]), the cost of our scheme is only 25 times more expensive than a 128-bit AES block cipher evaluation (AES is considered as very energy-efficient); i.e., the computational cost of our scheme is equivalent to encrypting a 3200-bit message using AES.

Moreover, we mention two simple ways to improve the efficiency of our scheme. First, note that generating the necessary column of G need only be done *once* by each node; that is, node i can generate $G(i)$ once (at the outset of the key-establishment phase) and then broadcast this column to each node with whom it desires to establish a common key. This reduces the *amortized* cost of establishing a key to only λ multiplications (this is an improvement if Horner's rule is not used above); this can also be further optimized if it is expected that nodes will need to compute a large number of shared keys. Second, to derive 64-bit keys it is not necessary work over a single field $GF(q)$ with $|q| \geq 64$; instead, one can define the key as the concatenation of four "sub-keys" that each lie in a field $GF(q)$ with $|q| \geq 16$. (For example, a single key space over $GF(2^{64})$ can be mapped to four independent key spaces over $GF(2^{16})$. This assumes $2^{16} > N$.) This will be more efficient since 4λ multiplications in a 16-bit field are more efficient than λ multiplications in a 64-bit field. The key observation is that security is not affected by working over $GF(q)$ where q is "small"; this is because our security arguments are information-theoretic and do not rely on any "cryptographic hardness" of the field $GF(q)$.

7. 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

$$\begin{aligned} & 1-(1-\frac{\tau}{\omega})(1-\frac{\tau}{\omega-1})\cdots(1-\frac{\tau}{\omega-\tau+1}) \\ \geq \ & \frac{(N-1)}{nN}(\ln(N)-\ln(-\ln(P_c))). \end{aligned} \tag{15}$$

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.

7.1 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. 8.

7.2 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

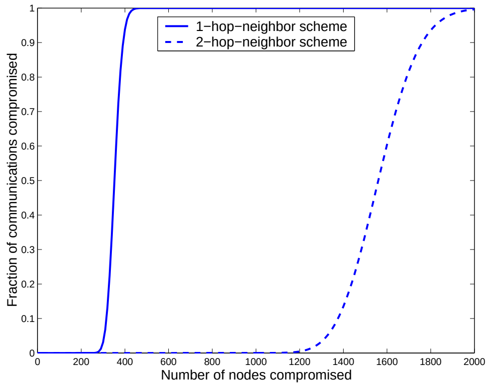


Fig. 8. Comparison: The left curve uses the 1-hop-neighbor scheme (with $\omega = 7$ and $\tau = 2$), and the right curve uses the 2-hop-neighbor scheme (with $\omega = 31$, and $\tau = 2$). Both figures achieve the same desirable global connectivity $P_c = 0.99999$.

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:

$$\text{Relative Overhead} = \frac{p'_h(1) + \sum_{\ell=2}^{L} 1.75\ell \cdot p'_h(\ell)}{\sum_{\ell=1}^{L} \ell \cdot p''_h(\ell)}, \tag{16}$$

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. 8 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.

8. CONCLUSIONS

We have proposed a framework in which to analyze the security of key pre-distribution schemes. We hope this framework will be useful to others working in this area. Much work remains to fully flesh out these definitions, and perhaps to achieve a more efficient construction of a secure pairwise authentication scheme without relying on the random oracle model.

We have also 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 analysis of the efficiency of our scheme. We have shown that when $p_{actual} \geq 0.33$, a node can (with very high probability) reach any neighbor within at most 3 hops. The computational requirements of our scheme are modest. Although our scheme involves modular multiplications, we have shown that the energy cost in establishing a key is (at worst) about the same as encrypting a 3200-bit message using AES. We also noted a number of ways to further optimize the computation of our scheme.

APPENDIX

A. CALCULATION OF $P_H(2)$ AND $P_H(3)$

We present our calculation of $p_h(2)$ and $p_h(3)$ in this appendix. We assume the distance between two nodes i and j is z.

A.1 Calculation of $p_h(2)$

The third node connecting these nodes must be in the overlapped region of the transmission range of node i and node j, as shown in Fig. 5.

As stated in Equation (14) the size of this overlapped region is:

$$A_{overlap}(z) = 2r^2 \cos^{-1}\left(\frac{z}{2r}\right) - z \cdot \sqrt{r^2 - \left(\frac{z}{2}\right)^2},$$

where r is the transmission range of each node.

Since, on the average, each node has n neighbors that are connected to it with wireless communication, the nodal density inside the transmission range is:

$$\rho = \frac{n}{\pi r^2}.$$

Thus, the total number of nodes in the overlap region is:

$$N_{overlap}(z) = \rho A_{overlap}(z).$$

Let $p_h(2, z)$ be the probability that i and j are not connected directly but there exists at least a common neighbor connecting them, given that the distance between i and j is z:

$$\begin{aligned} p_h(2,z) &= Pr\{[i \not\Leftrightarrow j] \cap [\exists \ell \in \mathcal{N}_i \cap \mathcal{N}_j \text{ s.t. } \ell \Leftrightarrow i \text{ and } \ell \Leftrightarrow j]\} \\ &= Pr\{i \not\Leftrightarrow j\} \cdot Pr\{\exists \ell \in \mathcal{N}_i \cap \mathcal{N}_j \text{ s.t. } \ell \Leftrightarrow i \text{ and } \ell \Leftrightarrow j | i \not\Leftrightarrow j\} \\ &= (1 - p_{actual})[1 - p_{2,1}(z)] \end{aligned}$$

where $\mathcal{N}_i$ and $\mathcal{N}_j$ represent the set of nodes that are in range of node i and j, respectively, $p_{2,1}(z)$ is the probability that none of the common neighbors of i and j is connected to both of them given that i and j are not connected, and $\Leftrightarrow$ means two nodes share at least one key space (connected). According to independence of key selections on each node,

$$p_{2,1}(z) = (p_{2,2})^{N_{overlap}(z)} \,,$$

where $p_{2,2}$ is the probability that a neighbor node, ℓ, of i and j is not connected to both of them given that i and j are not connected:

$$\begin{aligned} p_{2,2} &= 1 - \frac{\binom{\omega}{\tau}\binom{\omega-\tau}{\tau}}{\binom{\omega}{\tau}^3} \cdot \left\{ \binom{\omega}{\tau} - 2\binom{\omega-\tau}{\tau} + \binom{\omega-2\tau}{\tau} \right\} \\ &= 1 - \frac{\binom{\omega-\tau}{\tau}\left[\binom{\omega}{\tau} - 2\binom{\omega-\tau}{\tau} + \binom{\omega-2\tau}{\tau}\right]}{\binom{\omega}{\tau}^2}, \end{aligned}$$

where $\binom{\omega}{\tau}$ is the number of ways to select τ keys from ω key spaces for i, $\binom{\omega-\tau}{\tau}$ is the number of ways to select completely different τ keys for j, and $\binom{\omega}{\tau} - 2\binom{\omega-\tau}{\tau} + \binom{\omega-2\tau}{\tau}$ gives the number of ways to select keys for ℓ such that ℓ is connected to both i and j.

The PDF of z can be expressed as $f(z)$,

$$f(z) = \frac{\partial F(Z)}{\partial z} = \frac{\partial\left[Pr(Z \le z)\right]}{\partial z} = \frac{\partial}{\partial z}\left[\frac{\pi z^2}{\pi r^2}\right] = \frac{2z}{r^2},$$

thus, $p_h(2)$ is

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

where we substitute z with $y = \frac{z}{r}$.

A.2 Calculation of $p_h(3)$

$p_h(3)$ can be calculated with a similar method. We define $p_h(3, z)$ as the probability that 3 hops are needed to connect node i and node j, given that the distance between them is z ($z \le r$):

$$\begin{aligned} p_h(3, z) &= Pr\{[i \not\Leftrightarrow j] \cap [\forall \ell \in \mathcal{N}_i \cap \mathcal{N}_j \;\; \ell \text{ is not connected to both } i \text{ and } j\,] \cap \\ &\quad [\exists u \in \mathcal{N}_i \text{ and } v \in \mathcal{N}_j \text{ s.t. } u \Leftrightarrow i \text{ and } v \Leftrightarrow j \text{ and } u \Leftrightarrow v]\} \\ &= [1 - p_h(1) - p_h(2)][1 - p_{3,1}(z)] \end{aligned}$$

where $1 - p_{3,1}(z)$ is the probability that there exists at least a pair of nodes u and v connected to each other and connected to i and j separately, given that i and j are not directly connected, nor can they be connected through another common neighbor.

The exact calculation of $p_{3,1}(z)$ is complicated. We give an approximation as follows: For every neighbor v of node j, we find all the possible node u, which may satisfy $i \Leftrightarrow u \Leftrightarrow v \Leftrightarrow j$. We calculate the number of such pairs of (u, v).

Assuming that node v is at location (y, θ) from origin of j, the distance between node v and i is x:

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

Obviously, node u should reside in the shaded area in Fig. 6. The expected number of nodes residing in the small neighborhood of (y, θ) is $\rho y \cdot dy \cdot d\theta$. The number of nodes in the overlap region of circle i and circle v, $A_{overlap}(x)$, can be expressed $\rho \cdot A_{overlap}(x)$. So the total number of pairs of (u, v), given that the distance between i and j is z, is:

$$N_3(z) = \int_0^{2\pi} \int_0^r \rho^2 y \cdot A_{overlap}(x)\ dy\ d\theta$$

where, similar to Eq. (14), $A_{overlap}(x) = 2r^2 \cos^{-1}\left(\frac{x}{2r}\right) - x \cdot \sqrt{r^2 - (\frac{x}{2})^2}$.

So,

$$p_{3,1}(z) \ = \ (p_{3,2})^{N_3(z)} \tag{17}$$

where $p_{3,2}$ is the probability that for a pair of nodes $u \in \mathcal{N}_i$ and $v \in \mathcal{N}_j$, security connections cannot be made through path i, u, v, and j given that i and j are not directly connected or through a common neighbor. $p_{3,2}$ can be estimated[6] as follows:

$$\begin{aligned} \tilde{p}_{3,2} \ \approx \ & 1 - \frac{\binom{\omega}{\tau}\binom{\omega-\tau}{\tau}}{\binom{\omega}{\tau}^4} \cdot \sum_{a=1}^{\tau-1}\sum_{b=1}^{\tau-1}\sum_{c=1}^{\tau-\max(a,b)} \binom{\tau}{a}\binom{\tau}{b} \\ & \cdot \binom{\omega-2\tau}{c}\binom{\omega-2\tau-c}{\tau-a-c}\binom{\omega-2\tau-(\tau-a)}{\tau-b-c} \end{aligned} \tag{18}$$

where $\binom{\omega}{\tau}$ is the number of ways to select τ keys from ω key spaces for i, $\binom{\omega-\tau}{\tau}$ is the number of ways to select completely different τ keys for j, a represents the number common keys shared by u and i, b represents the number common keys shared by v and j, c represents the number common keys shared by u and v, $\binom{\omega-2\tau}{c}$ gives the number of ways to select the common keys different to i and j from the pool of key spaces, $\binom{\omega-2\tau-c}{\tau-a-c}$ is the number of ways to select the $\tau - a - c$ keys for u, and $\binom{\omega-2\tau-(\tau-a)}{\tau-b-c}$ gives the number of ways to select the $\tau - b - c$ keys for v.

Based on the distribution of z, $p_h(3)$ is:

$$p_h(3) \ \approx \ \int_0^r \frac{2z}{r^2}[1 - p_h(1) - p_h(2)]\left[1 - (\tilde{p}_{3,2})^{N_3(z)}\right] dz.$$

We substitute x, y, and z with $x' = \frac{x}{r}$, $y' = \frac{y}{r}$, and $z' = \frac{z}{r}$. We further simplify our notation by dropping the primes from these variables. Thus,

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

[6]Eq. (18) is an approximation because the probability is obtained by assuming only that node i and j are not connected.

REFERENCES

AKYILDIZ, I. F., SU, W., SANKARASUBRAMANIAM, Y., AND CAYIRCI, E. 2002. A survey on sensor networks. *IEEE Communications Magazine 40,* 8 (August), 102–114.

ANDERSON, R. AND KUHN, M. 1996. Tamper resistance - a cautionary note. In *Proceedings of the Second Usenix Workshop on Electronic Commerce*. 1–11.

BELLARE, M. AND ROGAWAY, P. 1993. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*. 62–73.

BLOM, R. 1985. 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.

BLUNDO, C., SANTIS, A. D., HERZBERG, A., KUTTEN, S., VACCARO, U., AND YUNG, M. 1993. Perfectly-secure key distribution for dynamic conferences. *Lecture Notes in Computer Science 740*, 471–486.

CARMAN, D. W., KRUUS, P. S., AND MATT, B. J. 2000. 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.

CHAN, H., PERRIG, A., AND SONG, D. 2003. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*. Berkeley, California, 197–213.

DU, W., DENG, J., HAN, Y. S., AND VARSHNEY, P. K. 2003. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proceedings of the 10th ACM conference on Computer and communications security*.

ERDŐS AND RÉNYI. 1959. On random graphs I. *Publ. Math. Debrecen 6*, 290–297.

ESCHENAUER, L. AND GLIGOR, V. D. 2002. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security*.

KAHN, J. M., KATZ, R. H., AND PISTER, K. S. J. 1999. Next century challenges: Mobile networking for smart dust. In *Proceedings of the 5th Annual ACM/IEEE Internation Conference on Mobile Computing and Networking (MobiCom)*. 483–492.

LIU, D. AND NING, P. 2003. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM conference on Computer and communications security*.

MACWILLIAMS, F. J. AND SLOANE, N. J. A. 1977. *The Theory of Error-Correcting Codes*. New York, NY: Elsevier Science Publishing Company, Inc.

MALKHI, D., REITER, M., WOOL, A., AND WRIGHT, R. N. 2001. Probabilistic quorum systems. *Information and Computation* 2 (November), 184–206.

NEUMAN, B. C. AND TSO, T. 1994. Kerberos: An authentication service for computer networks. *IEEE Communications 32,* 9 (September), 33–38.

PERRIG, A., SZEWCZYK, R., WEN, V., CULLAR, D., AND TYGAR, J. D. 2001. SPINS: Security protocols for sensor networks. In *Proceedings of the 7th Annual ACM/IEEE Internation Conference on Mobile Computing and Networking (MobiCom)*. Rome, Italy, 189–199.

PETERSON, W. W. 1972. *Error-Correcting Codes*, second ed. Cambridge, MA: Mass. Inst. Tech.