A Pairwise Key Predistribution Scheme for Wireless Sensor Networks

WENLIANG DU
Syracuse University
JING DENG
University of New Orleans
YUNGHSIANG S. HAN
National Taipei University
PRAMOD K. VARSHNEY
Syracuse University
and
JONATHAN KATZ and ARAM KHALILI
University of Maryland

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 nontrivial. 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. Predistribution 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.

In this paper, we provide a framework in which to study the security of key predistribution schemes, propose a new key predistribution 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

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].
Authors' addresses: W. Du, Department of Electrical Engineering and Computer Science, Syracuse University; email: wedu@ecs.syr.edu; J. Deng, University of New Orleans; Y. S. Han, Graduate Institute of Communication Engineering, National Taipei University, Sanshia, Taipai county, Taiwan, ROC; email: yshan@mail.ntpu.edu.tw; P. K. Varshney, Syracuse University; J. Katz and A. Khalili, University of Maryland, College Park, MD.

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 predistribution, security

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 ultrasmall 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, and so on. 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, that is, how to set up secret keys between communicating nodes.

This problem is known as the *key agreement* problem. Although the problem has been widely studied in general network environments, many schemes targeted at such environments are inapplicable to sensor networks due to the unique features of the latter. In particular, key agreement schemes for WSNs must satisfy the following requirements: (1) *Low energy consumption*: because sensor nodes are powered by batteries with limited power, a key agreement scheme should have low communication and computation costs. (2) *Low cost*: because sensor nodes are expected to be inexpensive, the associated hardware costs should be low. (3) *Low memory usage*: because sensor nodes have very limited memory, the memory requirements of the scheme should be low. (4) *Lack of trusted infrastructure*: sensor nodes are usually unattended and lack protection; therefore, none of the nodes (except possibly for a limited number of base stations) should be considered "trusted." (5) *Resilient against node capture*: the

[1]Wireless integrated network sensors, University of California. See: http://www.janet.ucla.edu/WINS.

resilience of the scheme should be high, where resilience refers to the percentage of communication links—not involving compromised nodes—which remain secure following compromise of a group of nodes. A scheme is “perfectly resilient” if the compromise of any node (or any group of nodes) does not compromise the security of any communication channels between noncompromised nodes.

Three types of key agreement schemes have been studied in general network environments: trusted-server schemes, public-key schemes, and key predistribution 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 one cannot generally assume that any trusted infrastructure is in place. Even if some base stations are available, relying on them for key agreement is inefficient because of the communication costs involved. *Public-key* schemes depend on asymmetric cryptography and typically assume some sort of public-key infrastructure which may not be present. Furthermore, the limited computational and energy resources of sensor nodes make it infeasible to use public-key algorithms in WSNs. A third approach to establish keys is via *predistribution*, where (secret) key information is distributed to all sensor nodes prior to deployment. Such schemes seem most appropriate for WSNs, and it is this type of scheme we consider here.

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 an a priori knowledge about the topology of the network does not exist. A number of key predistribution 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 predistribution scheme in which each sensor stores $N - 1$ keys (where N is the number of nodes in the network), each of which is known to only one other sensor node. This scheme guarantees perfect resilience because compromised nodes do not leak information about keys shared between two noncompromised nodes. Unfortunately, this scheme is impractical for sensors with an extremely limited amount of memory because N can be very large. Moreover, this scheme does not easily allow new nodes to be added to a preexisting sensor network because the existing nodes will not have the new nodes’ keys.

Recently, two random key predistribution schemes suited for sensor networks have been proposed. The first [Eschenauer and Gligor 2002] 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 *probabilistically* (this probability can be tuned by adjusting the parameters of the scheme). Eschenauer and Gligor note that this is not 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.

A generalization of this is the "q-composite" scheme [Chan et al. 2003] 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 additional communication links. The difference between this scheme and the previous one 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 predistribution 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 smaller 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 a 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 predistribution schemes for wireless sensor networks. Somewhat surprisingly, we find that prior definitions of security for key predistribution schemes are insufficient for typical applications; thus, we believe our framework is of independent interest and should prove useful for further work in this area.

Our key predistribution scheme combines the key predistribution scheme of Blom [1985] (see also Blundo et al. [1993]) with the random key predistribution methods discussed previously. (We review this scheme in detail in Section 3.) Blom's scheme allows *any* pair of nodes to compute a secret shared key. 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 trade-off is that, unlike the $(N - 1)$-pairwise-key scheme, Blom's scheme is no longer perfectly resilient against node capture. Instead, it has the following λ-secure property: as long as an adversary compromises no more than λ nodes, communication links between all noncompromised nodes remain secure. However, once an adversary compromises more than λ nodes, all 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 greater resilience. This threshold property of Blom's scheme is a desirable feature because one can set λ such that an adversary needs to attack a significant fraction of the network in order to achieve any payoff. However, increasing λ also increases the amount of memory required to store key information. 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 Blom's scheme) without using too much additional memory.

Roughly speaking, 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 predistribution schemes described previously [Chan et al. 2003; Eschenauer and Gligor 2002], we propose a new scheme using *multiple* key spaces. That is, we first construct ω spaces using Blom's scheme, and then have each sensor node carry key information from τ (with $2 \leq \tau < \omega$) randomly selected key spaces. Now (from the properties of the underlying Blom scheme), if two nodes carry key information from a common space they can compute a shared key. Of course, unlike Blom's scheme it is no longer certain that two nodes can generate a pairwise key; instead (as in previous random key predistribution schemes), we have only a probabilistic guarantee that this will occur. 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 predistribution schemes.

The remainder of this paper is organized as follows. Section 2 describes our proposed framework for analyzing the security of key predistribution schemes in terms of their effectiveness in establishing "secure (cryptographic) channels." We also show a simple method to convert any secure key predistribution scheme into a scheme for establishing such channels. Section 3 reviews Blom's key predistribution scheme which will be used as a building block of our main scheme, which is described in Section 4. Section 5 rigorously quantifies the resilience of our scheme against node capture, and compares our scheme with existing key predistribution schemes. Section 6 presents the communication and computational overheads of our scheme, and Section 7 describes some further improvements of our 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 are given in Section 5.

Blundo et al. [1993] proposed several schemes allowing any group of n parties to compute a common key which is perfectly secret with respect to any coalition of t other parties. When $n = 2$, their scheme is essentially equivalent to Blom's scheme (cf. Blundo et al. [1993]). Although both Blom's scheme (for $n = 2$) and the main scheme of Blundo et al. [1993] (for arbitrary n) match the known lower bound 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 are required to be able to compute a shared key and (2) the network is *perfectly* resilient to at most t captured nodes. By relaxing these requirements (slightly) and considering their *probabilistic* analogues, we obtain more memory-efficient schemes.

Perrig et al. [2001] proposed SPINS, 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 [2003], which was published at the same time as the conference version of this paper [Du et al. 2003]. Liu and Ning's approach is based on Blundo's (two-party) scheme, rather than on Blom's scheme as done here. Thus, Liu and Ning's scheme is essentially equivalent to the one shown here. However, as compared to Liu and Ning [2003], this paper provides a more thorough analysis of both the security and the communication overhead; we also introduce a rigorous framework (i.e., formal definitions of security) appropriate for analyzing key predistribution schemes for wireless sensor networks.

2. A SECURITY FRAMEWORK FOR KEY PREDISTRIBUTION SCHEMES

Before describing our primary scheme in detail, we first propose a framework in which to analyze the security of key predistribution schemes in general. Our starting point is the following simple observation: the goal of a key predistribution 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. In particular, we show below that although previous schemes ensure 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), these schemes do *not* necessarily guarantee security if this key K_{ij} is then used to, for example, authenticate the communication between these nodes. This emphasizes the importance of precise definitions of security, as well as rigorous proofs in some well-defined model.

We develop our framework as follows: We first define key predistribution schemes, and then describe for such schemes a "basic" level of security. This definition captures the idea that an adversary should (except with low probability) be unable to determine the key shared by some pair of users, and roughly corresponds to the level of security considered by Eschenauer–Gligor and all subsequent work in this area. We then define a stronger notion which more accurately represents the level of security expected from key predistribution schemes when used in practice. For simplicity, we focus on the case of message authentication; our results easily extend to other examples such as symmetric-key encryption. Our definition in this case (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 *cryptographic key distribution*. We then show that a scheme meeting the "basic" notion of security is not necessarily a secure cryptographic key distribution scheme. On a positive note, we show a simple way to convert any scheme achieving the "basic" level of security to one which *is* a secure cryptographic key distribution scheme. Our definitions, as well as our results, are described here in a relatively informal fashion. Yet, it is 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 predistribution 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. (The key derivation stage is assumed to be deterministic, but it may potentially require interaction between nodes i and j.) Execution of this algorithm by node i (holding information k^i) is denoted as $\mathsf{Derive}(k^i, i, j)$; we always require the basic correctness condition $\mathsf{Derive}(k^i, i, j) = \mathsf{Derive}(k^j, j, i)$. Note that a pair of nodes i, j is not guaranteed to be able to establish a shared key $K_{ij} \neq \perp$. For any distinct i, j, we assume that the probability (over choice of master key S) that i and j can establish a shared key (i.e., that $\mathsf{Derive}(k^i, i, j) \neq \perp$) is equal to some fixed parameter p, and we refer to this p as the *connectivity probability* of the scheme.

A "basic" level of security is defined via the following game: First run an instance of the key predistribution scheme. An adversary is given $I = \{(i_1, k^{i_1}), \ldots, (i_t, k^{i_t})\}$ for t randomly selected nodes $\{i_1, \ldots, i_t\}$ (this I represents what the adversary learns after compromising t randomly selected nodes). The adversary must then output (i, j, K), where $i, j, \notin I$, and $K \in \{0,1\}^\ell$ represents its "guess" for the key K_{ij}. We say the adversary *succeeds* if its guess is correct, and denote its probability of success (conditioned on the master secret information S and the information I which is available to the adversary) as $\Pr[\mathsf{Succ} \mid S, I]$. We say a key predistribution scheme is (t, ϵ, δ)-secure if for any adversary we have

$$\Pr_{S,I}[\Pr[\mathsf{Succ} \mid S, I] \leq \epsilon] \geq 1 - \delta.$$

We remark that in analyzing the security of our scheme in Section 5.1, we set $\epsilon = 2^{-\ell}$ (essentially the best possible, since the keyspace is $\{0,1\}^\ell$) and then derive appropriate relations between t and δ.

Before introducing a notion of security which is more along the lines of what is desired in practice, we augment a key predistribution scheme with an additional *message authentication* algorithm Mac and *message verification* algorithm Vrfy. Now, once 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 cryptographic key distribution via the following game: First run an instance of the key predistribution scheme, and give $I = \{(i_1, k^{i_1}), \ldots, (i_t, k^{i_t})\}$ to an adversary as before. Additionally, the adversary can repeatedly make an unbounded 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 $K_{i'j'}$) and returns the resulting tag to the adversary. Finally, the adversary outputs $(i, j, m^*, \mathsf{tag}^*)$ and we say the adversary *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 $\overline{\mathsf{Mac}}(j, i, m^*)$. That is, success corresponds to the adversary's ability to "insert" a bogus message m^* which is accepted as valid by one of i, j even though neither node authenticated this message. (This definition is a straightforward "lifting" of the standard notion of security for message authentication [Bellare et al. 2000] to the multiparty setting.) As above, let $\Pr[\mathsf{Succ} \mid S, I]$ denote the adversary's probability of success conditioned on the values of S and I. Fixing[2] some time bound T, we say a scheme is a (t, ϵ, δ)-secure *cryptographic key distribution scheme* if, for any adversary running in time T we have

$$\Pr_{S,I}[\Pr[\mathsf{Succ} \mid S, I] \leq \epsilon] \geq 1 - \delta.$$

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 predistribution scheme secure in the basic sense need not be a cryptographic key distribution 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., nonnegligible) 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 cryptographic key distribution 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 an information-theoretic sense), a formal proof that the scheme meets the requirements of a cryptographic key distribution scheme will not be possible. In fact, dependence between keys generated by the various pairs of parties reflects a serious potential vulnerability, as this leaves open the possibility of *related-key attacks* on the message authentication code or the lower-level primitives (e.g., block ciphers) from which the MAC is constructed. The possibility of such related-key attacks also rules out the easy "fix" in which nodes prepend the identities of the sender/receiver to any authenticated messages; although this prevents the "repeated-key " attack discussed earlier, it does nothing to protect against related-key attacks.

Luckily, it is simple to derive cryptographic key distribution schemes from key predistribution schemes in the *random oracle model* [Bellare and Rogaway

[2]We may also let T be a parameter of the definition, but for simplicity have not done so.

1993]. Let K_{ij} be the key derived by nodes i and j in some key predistribution 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*. 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, δ)-secure in the basic sense, and if the probability of forgery for the MAC is ϵ' (for an adversary running in time T), then the modified scheme is a $(t, q_h \cdot \epsilon + \binom{n}{2} \cdot \epsilon', \delta)$-secure cryptographic key distribution scheme, where q_h is a bound on the number of random oracle queries (i.e., hash function evaluations) made by an adversary. The proof is straightforward and is omitted here.

Since one may always convert any secure key predistribution scheme into a cryptographic key distribution 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 PREDISTRIBUTION SCHEME

Blom proposed a key predistribution method that allows any pair of nodes in a network to be able to derive a pairwise secret key [Blom 1985]. It has the property that as long as no more than λ nodes are compromised, all communication links of noncompromised nodes remain secure (we refer to this as being "λ-secure"); using the terminology of the previous section, the scheme is $(\lambda, 2^{-\ell}, 0)$-secure, where ℓ is the length of the shared key. We now briefly describe Blom's scheme (we have made some slight modifications to the scheme in order to make it more suitable for sensor networks, but the essential features remain unchanged).

We assume some agreed-upon $(\lambda + 1) \times N$ matrix G over a finite field $GF(q)$, where N is the size of the network and $q > N$. This matrix G is public information and may be shared by different systems; even adversaries are assumed to know G. During the key generation phase 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 must be kept secret and should not be disclosed to adversaries or to any sensor nodes (although, as will be discussed, one row of $(D \cdot G)^T$ will be disclosed to each sensor node). Because D is symmetric, it is easy to see that

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

that is, $A \cdot G$ 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 the ith row and jth column of K. The idea is to use K_{ij} (or K_{ji}) as the pairwise key between node i and node j. Figure 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 predistribution scheme,

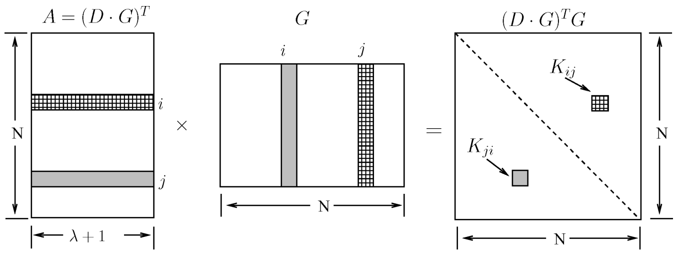


Fig. 1. Generating keys in Blom's scheme.

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]

Then, when nodes i and j need to establish pairwise key, they first exchange their columns of G and then 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 (not including i and j) have any information about K_{ij} or K_{ji}.

3.1 An Example of a Matrix G

We show an example of a matrix G which can be used in the above scheme. Recall 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 designed as follows [MacWilliams and Sloane 1977]:

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

Since s is primitive, $s^i \neq s^j$ if $i \neq j \bmod q$. Since G is a Vandermonde matrix and $q > N$, it can be shown that any $\lambda + 1$ columns of G are linearly independent [MacWilliams and Sloane 1977]. This matrix G has the nice property

[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.

that its columns can be generated by an appropriate power of the primitive element s. That is, to store the kth column of G at node k we need only store the seed s^k at this node which can then regenerate the column when needed. Other trade-offs between memory usage and computational complexity will be discussed later in the paper.

4. A MULTIPLE-SPACE KEY PREDISTRIBUTION SCHEME

Blom's scheme achieves optimal resilience at the expense of relatively large memory requirement. Here, we demonstrate a scheme which achieves good—although not optimal—resilience but which offers the advantage of requiring much lower memory usage. 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., an edge exists between any two nodes). Although such connectivity is desirable, it is not necessary. To achieve our goal of allowing any two nodes to communicate, all we need is a *connected* graph. By relaxing the requirement in this way, we achieve a scheme requiring much less storage.

Before we describe our proposed scheme, we define a *key space* (or *space* in short) as a matrix D as defined in the previous section. (The matrix G will be fixed.) We say a node *holds key space D* if the node stores the secret information generated from (D, G) using Blom's scheme. Note that two nodes can calculate pairwise key if they hold a common key space.

4.1 Key Predistribution Phase

During the key predistribution phase, we assign information to each node such that after deployment neighboring sensor nodes can establish a shared secret key with high probability. Assume that each sensor node has a unique identity ranging from 1 to N. Our key generation/distribution phase consists of the following steps:

Step 1: *Generating a 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 $q > N$), and then construct a matrix G of size $(\lambda + 1) \times N$ as discussed in the previous section. (Here, λ is parameter whose function will be discussed later.) Let $G(j)$ represent the jth column of G. Our goal is to provide $G(j)$ to node j. However, as discussed in Section 3, although $G(j)$ contains $(\lambda + 1)$ elements, each sensor only needs to store a "seed" (i.e., a single field element which is the second entry of the desired column) which can be used to regenerate $G(j)$. 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 keyspaces.* We generate ω random, symmetric matrices $D_1, \ldots, D_\omega$ of size $(\lambda + 1) \times (\lambda + 1)$. 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 per node.* For each node, we randomly select τ ($2 \leq \tau < \omega$) distinct key spaces from the ω possible choices. For each space D_i selected by node j, we store the jth row of A_i at this node. This information is secret; under no circumstance should a node send this information to any other node. Using Blom's scheme, two nodes can establish a common secret key if they both hold a common key space.

Since A_i is an $N \times (\lambda + 1)$ matrix, $A_i(j)$ contains $(\lambda + 1)$ 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 shared secret keys which will ultimately be generated, the memory usage of each node is $(\lambda + 1)\tau$ times the length of the key (we do not count the seed used to regenerate $G(j)$, since this seed may also serve as the node identity).

4.2 Key Agreement Phase

After deployment, each node needs to discover whether it shares a key 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 used to generate the appropriate column of G (as mentioned earlier, we could also let this be equal to the node identity, in which case this step is not needed).

Assume that nodes i and j are neighbors, and have sent the above broadcast messages. If they determine that they share a common space, say D_c, they can compute a 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 $K_{ij} = K_{ji}$ between nodes i and j can be computed in the following manner by these two nodes, respectively:

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

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 network. A key-sharing graph $G_{ks}(V, E)$ is defined 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 share 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 can 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 $i, v_1, \dots, v_t, j$. To establish a common secret key between i and j, node i first generates a random key K. Then i sends

[4]If we do not wish to disclose the indices of the spaces each node carries, we can use a challenge-response technique instead [Chan et al. 2003].

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 the 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 values for ω and τ such that the graph G_{ks} is connected with high probability? We use the following three-step approach, 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 [Chan et al. 2003; Eschenauer and Gligor 2002].

Step 1: *Computing required local connectivity.* Let P_c be the probability that the key-sharing graph is connected. We refer to this as the *global connectivity*. We let *local connectivity* p refer to the probability of two neighboring nodes sharing at least one space; that is, the probability that two neighboring nodes 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 threshold value called the *required local connectivity*, and denoted by p_{required}.

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

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

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 in G_{ks} should be at least d as calculated above, the required local connectivity p_{required} can be estimated as

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

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 two neighboring nodes sharing at least one key space (which is the same as the probability that they can establish a common key). Since

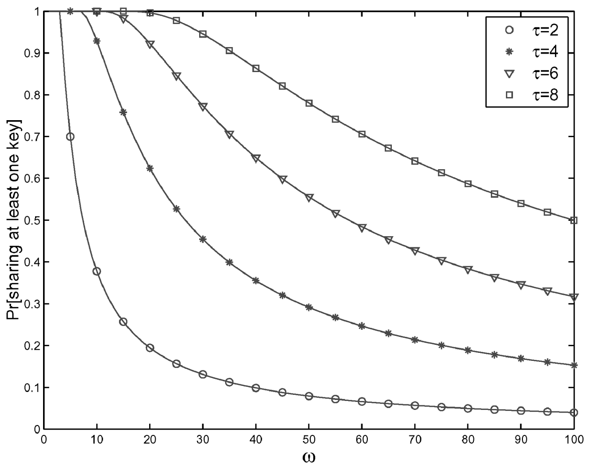


Fig. 2. Probability of two nodes sharing a key when each node hold τ key spaces chosen randomly from a set of ω key spaces.

$p_{\text{actual}} = 1 - \Pr(\text{two nodes do not share any space})$, we have

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

Values of p_{actual} have been plotted in Figure 2 for $\tau = 2, 4, 6, 8$ and ω varying from τ to 100. For example, one can see that when $\tau = 4$, the value of ω must be at most 25 in order to achieve local connectivity $p_{\text{actual}} \geq 0.5$.

The collection of sets of spaces assigned to each sensor forms a probabilistic quorum system [Malkhi et al. 2001]; the goal is for two sensors to have a space in common with high probability. Next we show that if $\tau \geq \sqrt{\ln \frac{1}{1-p_{\text{actual}}}}\sqrt{\omega}$, 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 observed in Figure 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,

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

Accordingly, to achieve a desired p_{actual} for a given ω we must have

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

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

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

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

Step 4: *Computing memory usage.* For each selected space in Blom's scheme, a node needs to carry $\lambda + 1$ field elements; Hence the total memory usage m for each node is

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

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

5. SECURITY ANALYSIS

We evaluate the multiple-space key predistribution 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. In our analysis, we assume that the adversary has no a priori knowledge of the keys carried by each sensor and we therefore model the attacker as compromising random nodes.[5]

5.1 Probability of At Least One Space Being Broken

We define our unit of memory 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 the ith key space is compromised (for $i \in \{1, \ldots, \omega\}$), let $\mathcal{C}_x$ be the event that x nodes are compromised in the network, and set $\theta = \frac{\tau}{\omega}$. We have

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

Applying the union bound, we obtain

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

[5]This assumption is reasonable due to the randomness in the key selection process, especially if we assume that a challenge-response technique is used to establish keys (cf. footnote 4).

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

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

Therefore,

$$\Pr(\text{at least one space is broken} \mid \mathcal{C}_x) \;\le\; \sum_{i=1}^{\omega} \Pr(\mathcal{S}_i \mid \mathcal{C}_x) \;=\; \omega \cdot \Pr(\mathcal{S}_1 \mid \mathcal{C}_x). \quad (6)$$

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

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

Combining inequality (6) and Eq. (7), we thus obtain the following upper bound:

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

We plot both simulation and analytical results in Figure 3. From the figure, the two results match each other closely, meaning that the union bound works quite well in the scenarios we discuss. Figure 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.

5.2 The Fraction of Compromised Network Communication

To better understand the resilience of our key predistribution scheme, we explore the effect of the capture of x sensor nodes by an adversary on the security of the rest of the network. In particular, we calculate 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 note that after the

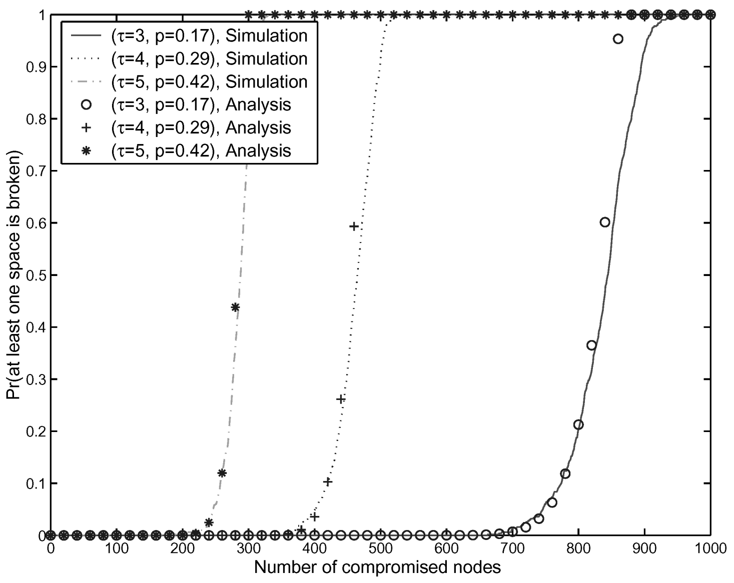


Fig. 3. The probability of at least one key space being compromised by the adversary when the adversary has captured x nodes ($m = 200$, $\omega = 50$). The value p in the figure represents p_{actual}.

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 communication 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 let K be the communication key used for this link. Let S_i denote the ith key space, and let $\mathcal{B}_i$ represent the joint event that K belongs to S_i and S_i is compromised. We use the notation $K \in S_i$ to represent that "key K was derived using S_i." The probability of c being compromised given the compromise of x other nodes 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 uses only one key, events $\mathcal{B}_1, \ldots, \mathcal{B}_\omega$ are mutually exclusive. Therefore,

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

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 events $\mathcal{C}_x$ and (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 using Eq. (7). The probability that K belongs to space S_1 is the probability that link c uses a key from space S_1. Since key spaces are assigned uniformly from the ω possibilities, we have

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

Therefore,

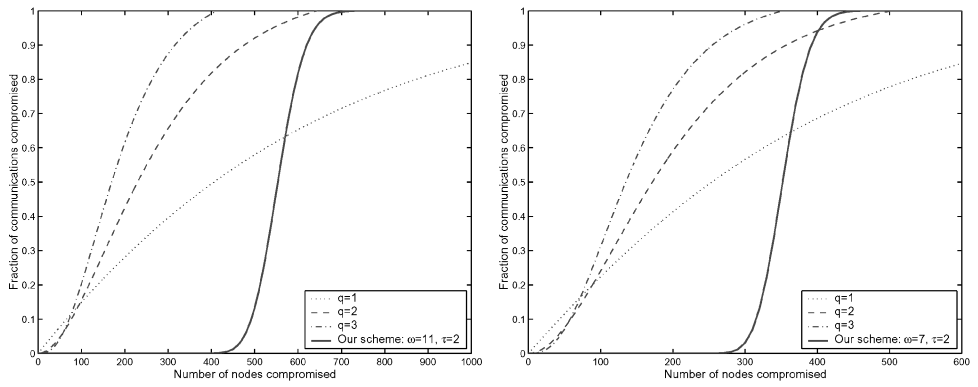
$$\begin{aligned}\Pr(c \text{ is broken} \mid \mathcal{C}_x) &= \omega \cdot \Pr(\mathcal{B}_1 \mid \mathcal{C}_x) \\ &= \omega \cdot \frac{1}{\omega} \cdot \Pr(S_1 \text{ is compromised} \mid \mathcal{C}_x) \\ &= \Pr(S_1 \text{ is compromised} \mid \mathcal{C}_x) \\ &= \sum_{j=\lambda+1}^{x} \binom{x}{j}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

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

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

5.2.1 *Comparison to Previous Work.* We first consider the compromise of links in the key-sharing graph. Figure 4 compares our scheme with the Chan–Perrig–Song scheme (for $q = 2, 3$) and the Eschenauer–Gligor scheme (i.e., with $q = 1$). The figure clearly shows the advantages of our scheme. Taking as an example the case in which $m = 200$ and $p_{\text{actual}} = 0.33$, in both the Chan–Perrig–Song and Eschenauer–Gligor schemes an adversary needs to compromise less than 100 nodes in order to compromise 10% of the links in the key-sharing graph. 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 an adversary for small-scale network breaches. We remark that although Chan et al. [2003] propose improving the security of their scheme using multipath key reinforcement, the same technique can be applied to our scheme to improve the security as well; we leave further comparison to our future work.

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).

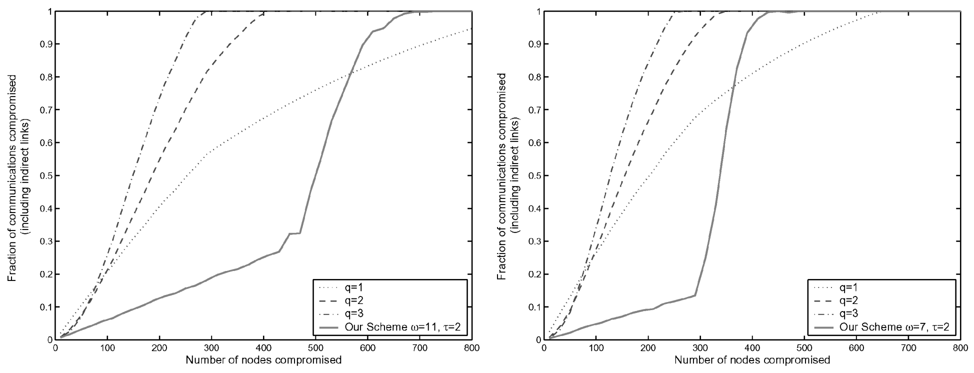


(a) $m = 200, p_{\text{actual}} = 0.33$ (b) $m = 200, p_{\text{actual}} = 0.5$

Fig. 4. Fraction of compromised links (in the key-sharing graph) between noncompromised nodes, after an adversary has compromised x random nodes. Here, m is the memory usage of the scheme and p_{actual} denotes the probability that any given pair of nodes can directly establish a pairwise key.

In Figure 4, we have only considered the security performance of our key predistribution scheme when two neighboring nodes can directly compute a shared key. Since the local connection probability is less than 1, two neighboring nodes might need to use a multihop path to set up a shared key (as discussed in Section 4). We refer to the secure channel established in this way as an *indirect link*. When any node or link along the multihop path used to establish an indirect link is compromised, the indirect link itself is also compromised. Our analysis in Figure 4 does not take such indirect links into account.

Due to the complexity of the analysis in this case, we used computer simulations to compare the resilience of our scheme in this case to previous schemes. We simulated a sensor network with $n = 1000$ nodes where indirect links were assumed between any pair of nodes where a direct link did not exist (the indirect link was assumed to be set up over the shortest existing path within the key-sharing graph); all other system parameters are the same as in Figure 4. We randomly picked x sensor nodes and considered them to be compromised. We then counted the number of secure links (including indirect links) that are compromised due to this capture. The results of our simulation are shown in Figure 5. From the figure, we see that our scheme is still significantly better than the Eschenauer–Gligor and Chan–Perrig–Song schemes. However, in all these schemes, the fraction of communication links compromised when indirect links are taken into account increases more quickly. This is due to the fact that, when considering indirect links, some of the intermediate nodes and links that help to establish the indirect links might be compromised, leading to the compromise of a portion of the indirect links. This also explains why the fraction of compromised links when $p_{\text{actual}} = 0.33$ is slightly higher than when $p_{\text{actual}} = 0.5$, as there are more indirect links in the former scenario than in the latter scenario.



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

Fig. 5. Fraction of compromised links (including indirect links) between noncompromised nodes, after an adversary has compromised x random nodes.

5.2.2 *Further Analysis.* Even though Eq. (9) can be used for numerical computation, it is too complex to allow a closed-form analytical result expressing the relationship between x, m, ω, and τ. The results in Figure 4 indicate that there is a small range of x in which the fraction of compromised links increases exponentially with respect to x. Here, we develop an analytical estimate of this range. It should be noted that Eq. (9) is the tail of a binomial distribution. Therefore, using known bounds on the tail of a binomial distribution [Peterson 1972] we can derive the following theorem whose proof is given in Appendix A.

THEOREM 5.1. *Assume that $\lambda = \frac{m}{\tau} \gg 1$, so that $\lambda + 1 \approx \lambda$. Define the entropy function of y, for $0 \leq y \leq 1$, as $H(y) = -y \ln y - (1 - y) \ln(1 - y)$ and let $H'(y) = dH(y)/dy$. Then 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

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

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 much less than $\frac{m\omega}{\tau^2}$. For example, when $m = 200$, $\tau = 2$, $\omega = 11$, and x is 25% less than $\frac{m\omega}{\tau^2}$ (i.e., $x = 0.75 \cdot \frac{m\omega}{\tau^2} = 413$), then the upper bound is $e^{-5.089} = 0.006$, which is 2 orders of magnitude smaller than 1. Hence, $\frac{m\omega}{\tau^2}$ can be used as an estimate (upper bound) of the value of x for which 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 Figure 4 verify that this estimate 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.

When the average number of neighbors of each sensor is decreased, Eq. (2) shows that the value of p_{required} increases. For a network of size N with desired global connectivity P_c, the value of p_{actual} must be increased in order to guarantee that the whole network is connected. However, the resilience of our scheme is weakened due to larger p_{actual}. In the following, we give a simple sufficient condition on n (the average number of neighbors per node) such that our scheme is "useful"; that is, has better resilience (for a given amount of memory) than Blom's scheme. That is, we want to find the minimum value of n which guarantees $\omega/\tau^2 > 1$. As derived in Section 4.3,

$$p_{\text{actual}} \geq 1 - e^{-\frac{\tau^2}{\omega}}.$$

Thus, when $1 - e^{-\frac{\tau^2}{\omega}} \geq p_{\text{required}} = d/n$ then $p_{\text{actual}} \geq p_{\text{required}}$. It is easy to derive that whenever $n \geq \lceil \frac{d}{1-e^{-1}} \rceil$, the requirement $p_{\text{actual}} \geq p_{\text{required}}$ can be satisfied while simultaneously achieving $\omega/\tau^2 > 1$. For example, with $N = 1000$ and $p_c = 0.9999$, having $n = \lceil \frac{d}{1-e^{-1}} \rceil = 26$ neighbors per node (on average) implies that our scheme is "useful."

6. OVERHEAD ANALYSIS

6.1 Communication Overhead

According to our previous discussions, the probability p_{actual} that two neighboring 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. Our analytical approach is similar to that given in Chan et al. [2003].

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 Figure 6. The size of this

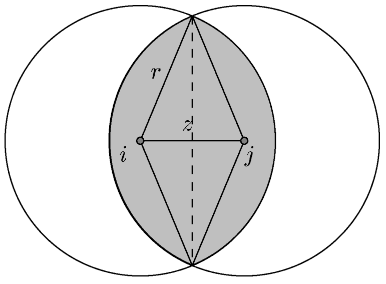


Fig. 6. Overlap region $A_{\text{overlap}}(z)$.

overlap region is

$$A_{\text{overlap}}(z) = 2r^2 \cos^{-1}\left(\frac{z}{2r}\right) - z \cdot \sqrt{r^2 - \left(\frac{z}{2}\right)^2}, \tag{12}$$

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

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

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 one common neighbor connecting them, given that the distance between i and j is z:

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

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 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 Figure 7.

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

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

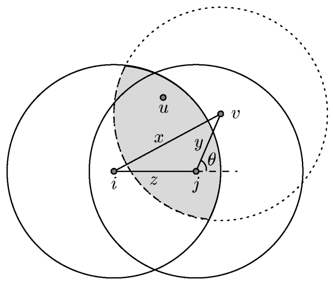


Fig. 7. Overlap region for $p_h(3)$.

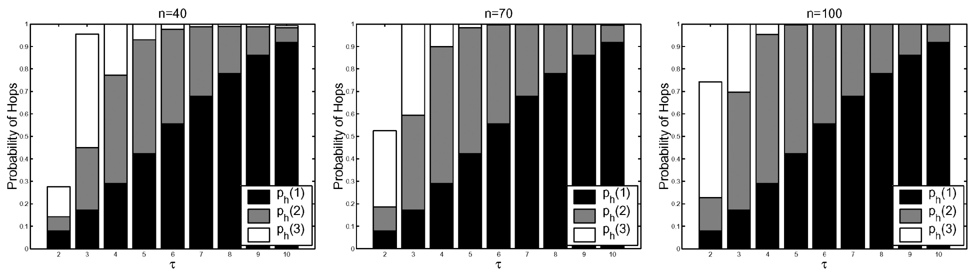


Fig. 8. Distribution of the number of hops required to connect neighbors ($\omega = 50$).

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

where

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

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

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

We plot the values of $p_h(1)$, $p_h(2)$, and $p_h(3)$ in Figure 8. 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 two 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 G is a Vandermonde matrix, the dominating computational cost of 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 multiplications come from the inner product of the corresponding row of $(DG)^T$ with this column of G. Note that this can be easily reduced to only λ multiplications using Horner's rule for

Table I. Time (ms) for Computing a 64-bit Secret Key ($\lambda = 50$)

	Four 16-bit Keys	Two 32-bit Keys	One 64-bit Key
Time (ms)	8.94	14.45	25.67

polynomial evaluation. (Although $O(\lambda)$ additions in $GF(q)$ are also necessary, these are dominated by the field multiplications.)

A natural choice is to work with fields of characteristic 2 (i.e., fields of the form $GF(2^k)$) both because multiplications in this field are rather efficient and also because elements in such fields naturally map to bit strings which can then be used as cryptographic keys. We observe that to derive a 64-bit key it is not necessary to work over $GF(2^k)$ with $k \geq 64$; instead, one can define the key as the concatenation of multiple "subkeys" each of which lie in a smaller field. As an example, a 64-bit key can be composed of four 16-bit keys. In general, this will lead to improved efficiency since, continuing with the above example, 4λ multiplications in $GF(2^{16})$ are more efficient than λ multiplications in $GF(2^{64})$. 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 $GF(q)$. The only requirement is that we work in a field $GF(q)$ with $q > N$, where N is the number of nodes in the network.

We implemented our key predistribution scheme on MICAz sensor nodes [Crossbow Technoloy, Inc.]. The time of computing a 64-bit key when $m = 200$ and $\tau = 4$ (i.e., $\lambda = 50$) is described in Table I for various underlying fields. Note that when working over $GF(2^{16})$ the total number of multiplications is $\frac{64}{16} * \lambda = 200$, while when working over $GF(2^{32})$ the total number of multiplications is $\frac{64}{32} * \lambda = 100$. We list the performance for computations using 16-bit, 32-bit, and 64-bit multiplications. The performance results indicate that moving to smaller fields does improve the performance. More importantly, the results show that our key predistribution scheme is quite practical: if we use four 16-bit sub-keys as a 64-bit key, a sensor can compute over 100 such keys within one second.

7. IMPROVING SECURITY USING TWO-HOP NEIGHBORS

In this section we describe a way to further improve the security of our key predistribution scheme, following Chan et al. [2003]. Using inequality (4), we have

$$1 - e^{-\frac{\tau^2}{\omega}} \geq \frac{(N-1)}{nN}[\ln(N) - \ln(-\ln(P_c))]. \tag{13}$$

Note 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.

One can increase n by increasing the communication range of a node, but this also increases the energy consumption. Another approach 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 of both of them. Nodes i and j then exchange information as in the one-hop case, except that this is done via I. Then, i and j find a common key space, and compute their secret key as before. Nodes i and j can then encrypt any future communication between them using this key. Although all future communication still needs to pass through an intermediate node, the intermediate node cannot decrypt the message if it does not carry the key space shared by i and j.

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; that is, 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 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 only difference is that now some edges in the graph represent a channel between two-hop neighbors, and thus message 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 a factor of four. Therefore, the expected number of neighbors n' for each node in G_{eks} is about four times as large in G_{ks}. According to Eq. (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, decreasing the desired p_{actual} means that ω can be increased. For example, assuming network size $N = 10,000$, connectivity probability $P_c = 1 - 10^{-5}$, and fixing $\tau = 2$, we need to select $\omega = 7$ for the G_{ks}-based key agreement scheme; however, using the G_{eks}-based scheme, we can select $\omega = 31$. The security of the latter scheme is improved significantly. Using Eq. (9), we plot the fraction of compromised links for the above two cases in Figure 9.

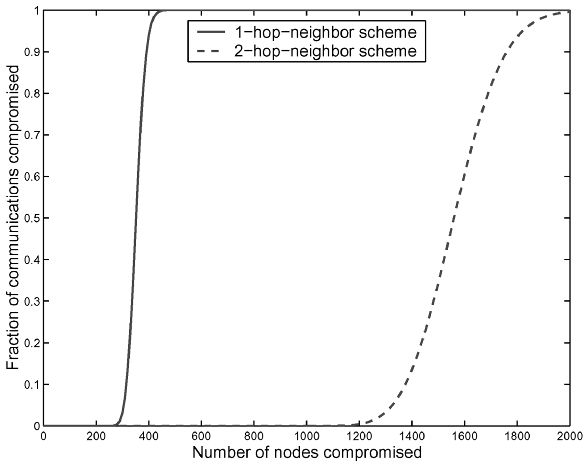


Fig. 9. Fraction of compromised links in the key-sharing and extended key-sharing graphs. The left curve uses the one-hop-neighbor scheme (with $\omega = 7$ and $\tau = 2$), and the right curve uses the two-hop-neighbor scheme (with $\omega = 31$, and $\tau = 2$). Both figures achieve the same global connectivity probability $P_c = 0.99999$. Note that the resilience only depends on the values of ω and τ, while the connectivity probability depends on whether a one-hop or two-hop scheme is used.

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-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, however, that in practice one can achieve better than this by selectively choosing one-hop edges when they exist). 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 (i.e., only using direct neighbors); assume the maximum length of the shortest path between two neighbors is L. Then the ratio between the overhead of the two-hop-neighbor scheme and that of the basic scheme can be estimated using the following formula:

$$\text{Relative overhead} = \frac{p'_h(1) + \sum_{\ell=2}^{L} 1.75\ell \cdot p'_h(\ell)}{\sum_{\ell=1}^{L} \ell \cdot p''_h(\ell)}, \tag{14}$$

where we do not need to multiply first term by 1.75 since if two neighbors share a common key, the path between them is never a two-hop edge. As an example, the overhead ratio of the two schemes used in Figure 9 is 3.18: namely, with 3.18 times more overhead, the resilience is improved by a factor of 4. The communication cost discussed here occurs only during the key setup phase, so it is a one-time cost.

8. CONCLUSIONS

We have proposed a framework in which to analyze the security of key predistribution schemes, and we expect this framework will be useful to others working in this area. We have also presented a new pairwise key predistribution scheme for wireless sensor networks. Our scheme has a number of appealing properties. First, our scheme is scalable and flexible, and nodes do not need to be deployed at the same time; they can be added after initial deployment, and still be able to establish secret keys with existing nodes. Compared to existing key predistribution 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 a network secured using our scheme, an adversary has to compromise five times as many nodes as he/she had to compromise in a network secured by the Chan–Perrig–Song or Eschenauer–Gligor schemes. Furthermore, we have also shown that network resilience can be further improved if we use multihop neighbors.

We have conducted a thorough analysis of the efficiency of our scheme. We have shown that when $p_{\text{actual}} \geq 0.33$, a node can (with very high probability) reach any neighbor within at most three hops. The computational requirements of our scheme are very modest, as demonstrated by our implementation on MICAz sensor nodes and resulting performance.

APPENDIX A: PROOF OF THEOREM 5.1

Assume $x \geq \lambda + 1$. According to the bound on the tail of a binomial distribution [Peterson 1972], Eq. (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

$$\sum_{j=\lambda+1}^{x}\binom{x}{j}\theta^{j}(1-\theta)^{x-j} \leq \alpha^{-\alpha x}(1-\alpha)^{-(1-\alpha)x}\theta^{\alpha x}(1-\theta)^{(1-\alpha)x}, \tag{15}$$

where $\alpha = \frac{\lambda+1}{x}$ and $\theta = \frac{\tau}{\omega}$. Since $\lambda = \frac{m}{\tau} \gg 1$, we have $\lambda + 1 \approx \lambda$. Consequently, $\alpha \approx \frac{\lambda}{x} = \frac{m}{\tau x}$. By taking the logarithm of the upper bound of inequality (15) 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,

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

giving the claimed result.

APPENDIX B: CALCULATION OF $P_H(2)$ AND $P_H(3)$

In the following, we assume the distance between two nodes i and j is z.

B.1 Calculation of $p_h(2)$

The third node connecting nodes i and j must be in the overlapped region of the transmission range of node i and node j, as shown in Figure 6. As stated in Eq. (12) the size of this overlapped region is

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

where r is the transmission range of each node. Since, on the average, each node has n neighbors within communication range, 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

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

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

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

where $\mathcal{N}_i$ and $\mathcal{N}_j$ represent the set of nodes in range of nodes 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. Since the choices of key spaces for each node are independent,

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

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. Also

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

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, denoted $f(z)$, can be expressed as

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

Thus, we have

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

where we replace z by $y = \frac{z}{r}$.

B.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 three 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 possible nodes u, which may satisfy $i \Leftrightarrow u \Leftrightarrow v \Leftrightarrow j$. We then calculate the number of such pairs of (u, v). Assuming that node v is at location (y, θ) (putting j at the origin), the distance x between nodes v and i is

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

Obviously, node u should reside in the shaded area in Figure 7. 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, $A_{\text{overlap}}(x)$, can be expressed as $\rho \cdot A_{\text{overlap}}(x)$. So the total number of pairs (u, v), given that the distance between i and j is z, is

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

where, similar to Eq. (12), $A_{\text{overlap}}(x) = 2r^2 \cos^{-1}\left(\frac{x}{2r}\right) - x \cdot \sqrt{r^2 - (\frac{x}{2})^2}$.

So,

$$p_{3,1}(z) = \left(p_{3,2}\right)^{N_3(z)}, \tag{16}$$

where $p_{3,2}$ is the probability that for a pair of nodes $u \in \mathcal{N}_i$ and $v \in \mathcal{N}_j$, secure connections cannot be made through path i, u, v, and j, given that i and j are not directly connected nor can they be connected through a common neighbor. $p_{3,2}$ can be estimated[6] as follows

$$\tilde{p}_{3,2} \approx 1 - \frac{\binom{\omega}{\tau}\binom{\omega-\tau}{\tau}}{\binom{\omega}{\tau}^4} \cdot \sum_{a=1}^{\tau-1}\sum_{b=1}^{\tau-1}\sum_{c=1}^{\tau-\max(a,b)} \binom{\tau}{a}\binom{\tau}{b} \cdot \binom{\omega-2\tau}{c}\binom{\omega-2\tau-c}{\tau-a-c}\binom{\omega-2\tau-(\tau-a)}{\tau-b-c}, \tag{17}$$

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 of common keys shared by u and i, b represents the number of common keys shared by v and j, c represents the number of 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, we have

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

replacing 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,

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

[6]Equation (17) is an approximation because the probability is obtained using only the fact that node i and j are not directly connected.

REFERENCES

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

ANDERSON, R. AND KUHN, M. 1996. Tamper resistance—A cautionary note. In *Proceedings of the 2nd Usenix Workshop on Electronic Commerce*. 1–11.

BELLARE, M., KILIAN, J., AND ROGAWAY, P. 2000. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences 61*, 3, 362–399.

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. In *Advances in Cryptology: Proceedings of EUROCRYPT 84*, T. Beth, N. Cot, and I. Ingemarsson, Eds. Lecture Notes in Computer Science, vol. 209, Springer-Verlag, Berlin, 335–338.

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

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

CROSSBOW TECHNOLOGY, INC. Available at http://www.xbow.com/.

DU, W., DENG, J., HAN, Y. S., AND VARSHNEY, P. 2003. A pairwise key predistribution scheme for wireless sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*. 42–51.

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*. 41–47.

KAHN, J., KATZ, R., AND PISTER, K. 1999. Next century challenges: Mobile networking for smart dust. In *Proceedings of the 5th Annual ACM/IEEE International 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*. 52–61.

MACWILLIAMS, F. AND SLOANE, N. 1977. *The Theory of Error-Correcting Codes*. Elsevier Science, New York.

MALKHI, D., REITER, M., WOOL, A., AND WRIGHT, R. N. 2001. Probabilistic quorum systems. *Information and Computation 170*, 2, 184–206.

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

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

PETERSON, W. W. 1972. *Error-Correcting Codes*, 2nd ed. MIT Press, Cambridge, MA.

Received September 2003; revised November 2004; accepted January 2005