Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2006, Article ID 91304, Pages 1–10
DOI 10.1155/WCN/2006/91304

A Robust on-Demand Path-Key Establishment Framework via Random Key Predistribution for Wireless Sensor Networks

Guanfeng Li,[1] Hui Ling,[1] Taieb Znati,[1] and Weili Wu[2]

[1] *Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260, USA*
[2] *Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083-0688, USA*

Received 2 October 2005; Revised 11 January 2006; Accepted 12 January 2006

Secure communication is a necessity for some wireless sensor network (WSN) applications. However, the resource constraints of a sensor render existing cryptographic systems for traditional network systems impractical for a WSN. Random key predistribution scheme has been proposed to overcome these limits. In this scheme, a ring of keys is randomly drawn from a large key pool and assigned to a sensor. Nodes sharing common keys can communicate securely using a shared key, while a path-key is established for those nodes that do not share any common keys. This scheme requires moderate memory and processing power, thus it is considered suitable for WSN applications. However, since the shared key is not exclusively owned by the two end entities, the established path-key may be revealed to other nodes just by eavesdropping. Based on the random-key predistribution scheme, we present a framework that utilizes multiple proxies to secure the path-key establishment. Our scheme is resilient against node capture, collusive attack, and random dropping, while only incurring a small amount of overhead. Furthermore, the scheme ensures that, with high probability, all path-keys are exclusively known by the two end nodes involved in the communication along the path.

1. INTRODUCTION

Recent advances in wireless technologies have led to a new generation of inexpensive sensors and actuators. Individually, these devices are resource-constrained and, as such, are only capable of a limited amount of processing and communication. When deployed in a large number, however, the coordinated effort of these networked devices bears promises for a significant impact, not only on science and engineering, but equally importantly on a broad range of civil and military applications, including health care, critical infrastructure protection, environmental and wildlife monitoring, crisis management, and military reconnaissance.

Harnessing the potential of wireless sensor networks, however, brings about a number of fundamental challenges, the most critical of which is security. It is frequently the case that sensors are deeply embedded into the environment or deployed in open areas, making them vulnerable to physical attacks and potentially compromising sensor nodes' security. Secure communication among sensors, during the response phase to an attack on a critical infrastructure, for example, is crucial for emergency responders to successfully coordinate their activities. Malicious information, injected by attackers during the response phase may hamper greatly the ability of first responders to communicate and share data. Cryptology methods are, therefore, needed to achieve secure communication among sensor nodes.

Since sensors will either have to be powered by small nonrenewable batteries, or by a modest amount of energy that can be harvested from the environment, developing energy-efficient cryptographic algorithms and methods is a critical issue in designing security protocols for wireless sensor networks. The sensors' resource constraints, coupled with their limited knowledge of the topology within which they are deployed, render public-key-infrastructure-(PKI) based schemes inappropriate for wireless sensor networks. Carman et al. pointed out that asymmetric cryptography algorithms, like 1024-bit RSA, consume at least two orders of magnitude more energy than symmetric cryptography algorithms, such as 1024-bit AES in [1]. Furthermore, symmetric-key cipher and hash functions execute between two to four orders of magnitude faster than their asymmetric counterparts. Similarly, trusted server-based cryptography systems, such as Kerberos, do not apply in WSNs, as these schemes require a trusted third party which is not always available in WSNs. Consequently, these schemes may not be scalable when a WSN involves thousands of sensor nodes. These constraints

leave designers of security protocols for WSNs with no choice but to use symmetric-key cryptographic systems.

In symmetric-key cryptographic systems, keys have to be installed onto sensors before deployment. Nodes then use shared keys to conduct secure communication. Two strategies can be used to distribute shared keys between sensors in WSNs. In the first strategy, all sensor nodes share the same session key, while in the second case each sensor node shares a unique key with each of the remaining $n-1$ sensors, where n is the total number of sensors in the WSN. The advantage of the first strategy stems from its low maintenance cost. In this strategy, however, the compromise of one single node may jeopardize the security of the entire network. The second strategy has potential to achieve perfect security even when a number of nodes are captured. In large WSNs, however, this approach requires installing $n-1$ keys in each sensor and, as such, may be prohibitive, given the limited memory size of a sensor node. Furthermore, sensors are likely to fail due to hardware faults or energy depletion caused by excessive communication. Consequently, in order to maintain the level of node density required to meet the quality of service requirement of the applications, new sensors may have to be injected into the existing network. The addition of these nodes further limits the applicability of the second approach, as it requires installing new keys into the existing sensors in order to facilitate communication between these sensors and the newly injected ones.

To overcome the shortcomings of the above strategies, a random key predistribution scheme has been proposed [2]. This scheme only requires a relatively small number of keys, in the order of ten to one hundred, to be installed onto each node, to achieve connectivity between pair of nodes with high probability. The link with two end nodes sharing keys is called secure link. Nodes that do not share a key set up a path-key, through negotiation, using paths formed by secure links. The major shortcoming of this scheme is during path-key establishment, communication between the end nodes is exposed to intermediate nodes along the path.

This path-key establishment problem has been introduced in [3]. Furthermore, it is shown that the risk of the path-key being revealed can be significantly decreased by using multiple node-disjoint secure paths to establish the path-key. However, the proposed scheme may incur too much extra overhead due to the necessity of discovering multiple node-disjoint paths between sources and destinations.

In this paper, we aim to set up a framework that uses multiple secure-one-hop paths instead of node-disjoint paths to enhance the security of path-key establishment. We present two efficient algorithms for discovering these intermediate hops (referred as proxy). It is shown both through analysis and simulation that our scheme can achieve a very high level of security, while simultaneously reducing the overhead.

The rest of this paper is organized as follows. Section 2 introduces related work. In Section 3, we describe our robust key establishment framework to secure the path-key using multiple proxies. Furthermore, we show how to discover such proxies in two algorithms. The security analysis and simulation results show that our scheme can achieve a high level of security. Conclusions are drawn in Section 4.

2. RELATED WORK

The random key predistribution scheme was first introduced by Eschenauer and Gligor [2]. Using this framework, different methods of key generation and distribution have been proposed to improve energy efficiency and security [4–7]. A q-composite random key predistribution scheme has been proposed which increases the security of key set-up in way such that an attacker has to capture a large number of nodes to compromise a communication, with high probability [4]. The authors also propose a multipath-key reinforcement scheme to update an existing link key to a unique key, thereby ensuring that the key is not used by any other sensor node. Although the scheme proposed in this paper uses multiple paths, it differs from the one proposed in [4] in that the proposed scheme achieves the same level of security, without the need to use node-disjoint paths. Computing node-disjoint paths is known to be NP-hard, and, therefore, may result in considerable communication overhead.

In the random key predistribution scheme proposed in [5], each node only needs to carry a fraction of the keys required by [2], while achieving the same level of security. This scheme has the potential to reduce memory usage and improve the network's resilience against node compromise. To achieve this goal, however, the scheme requires prior knowledge of sensor deployment within the WSN, which may not be readily available at any time. Furthermore, if the sensor nodes are moving, the network topology changes, thereby making prior knowledge deployment obsolete. Deployment knowledge was also used in [8] to improve key predistribution. The authors further exploited postdeployment knowledge to discard keys in a node where the keys are not shared with the node's actual neighbors to thwart node compromise attacks and vacate the precious memory space to be used by the loaded applications.

Another location-based key predistribution scheme based on deployment knowledge is described in [9]. A distinguishing property of this framework from the above schemes is that it does not require the knowledge of sensors' expected locations, but only requires to deploy sensor nodes in groups. Consequently the burden to deploy each sensor in the sensor network to the vicinity of its expected location is greatly reduced. However, since sensors are deployed in groups, node addition and key revocation in existing nodes are not easy for this network.

In [10] a seed-based key deployment strategy to discover shared keys, in a more energy-efficient manner, is described. All the keys in the key pool are indexed. Each node uses its ID as the seed and uses a pseudorandom function to generate the key indexes. It then loads the corresponding keys onto itself. This scheme requires more memory space as nodes have to store the associated indexes along with the keys. The scheme may also require additional computation, but no communication is required for two nodes to discover if they share a key between them.

The schemes described in [7] use a similar technique to discover shared keys. Although these new schemes save communication, they pose a security threat. After capturing a node, an attacker can gain additional advantage by selectively eavesdropping on nodes that are known to share keys with the captured one.

To prevent this attack, the key distribution strategy proposed in this paper adopts the original scheme described in [2]. Notice that the scheme described in [10] also sets up path-keys using different logic paths. However, the scheme cannot use the original shared key mechanism to discover these logic paths.

Both [4, 6] propose a scheme to support key authentication by generating unique pairwise keys. In [4], a node loads a set of node IDs and a unique pairwise key k is generated for each pair of nodes. Hence, if k is used to secure communication, both nodes are certain of their respective identities, since no other node pair can hold k. In [6] the random key predistribution is combined with Blom's key predistribution scheme [4] to achieve "λ-security." This level of security is achieved only if an adversary cannot compromise more than λ nodes; uncompromised nodes remain perfectly secure. When more than λ nodes are captured, the entire network may be compromised if just one key space is used.

While the security of random key predistribution schemes has made significant improvement, the path-key establishment problem has not yet been fully addressed [3]. In this paper, we propose to improve the path-key security using multiple secure proxies.

3. ROBUST PATH-KEY ESTABLISHMENT

We first review Gligor's work upon which our work is based and give an example to highlight the main idea. In his scheme, each node is installed with a key ring of m keys randomly drawn from a large key pool, P. This scheme requires moderate memory space for storing a key ring, and therefore can be used in a very large network. For example, if a key ring consists of 20 keys and is drawn from 1000 keys, theoretically it can support up to $\binom{1000}{20} = 2.4 \times 10^{19}$ nodes, and only requires 160 bytes assuming 64-bit key cryptography system. After being deployed, two nodes within transmission range exchange either key identifiers or challenges to discover common keys in their key rings. Then a common key is selected for secure communication between these two nodes. Node pairs without a common key establish a path-key through a secure path.

In the network depicted in Figure 1, it is assumed that shared keys have been discovered as illustrated by dashed links. According to this example, $N1$ shares a key with $N2$ but not with $N3$ or $N4$. When $N1$ wants to communicate with $N3$, it finds a secure path $N1 \rightarrow N2 \rightarrow N4 \rightarrow N3$ and sends a key K to $N3$ through the established path. K is encrypted with K_{12}, K_{24}, and K_{34}, respectively, as it travels from $N1$ to $N3$. Notice, however, that while the pairwise key K is supposed to be exclusively shared between $N1$ and $N3$, the need for successive decryptions and encryptions along the path causes the key to be exposed to the intermediate nodes

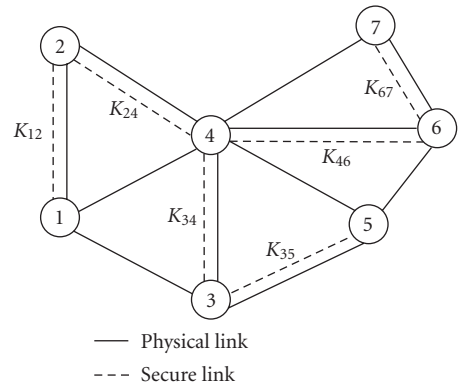

Figure 1: An example sensor network after shared key discovery.

$N2$ and $N4$. This may lead to potential security compromise if a node along the path is captured. This problem is referred as the "path-key establishment problem" in [3].

Another security concern about this framework is that the probability of any two nodes sharing keys is high. In the last example, the probability is 33.5%. If the key ring size is increased to 30, the probability will be over 60.5%. Any key in the key pool has a probability equal to (key ring size/key pool size) to be installed on one node. In a large WSN, if two nodes are using a shared key to talk to each other, chances are, there will be some nodes in the neighborhood that hold this shared key they are using. This situation demands that two nodes set up a path-key for a private communication even if they shared keys on their key rings.

Using multiple node-disjoint paths to secure the path-key establishment has been proposed to cope with the risk that compromising one node along the path leads to revealing the path-key [3]. A path-key K is broken down into k nuggets and sent along k node-disjoint paths. All nuggets are required to reconstruct K. Therefore, an attacker would have to capture at least one node along each path to obtain the key. However, as pointed out above, these key nuggets are exposed to each intermediate node along the routing path. In summary, this scheme has the following undesirable features.

(i) It involves a high level of overhead to find node-disjoint paths. Furthermore, in some cases, it may not be physically feasible to construct k node-disjoint paths.

(ii) Contrary to intuition, increasing the number of node-disjoint paths does not necessarily improve the level of security of the underlying path-key establishment scheme. This is because as the number of node-disjoint paths increases, so does the number of intermediate nodes. This in turn increases the chances of the path-key being exposed to adversaries.

To reduce the exposure of the key nugget along the path, the proposed scheme ensures that no more than one node along a path knows the key nugget. This node is referred to as a *proxy*. The proxy shares a key with each end node, respectively. Now that the key nugget is secured by the proxy,

it becomes feasible to relax the node-disjoint requirement of the k paths without increasing the vulnerability of the path-key. Furthermore, since these paths no longer require to be composed of secure links only, any physical path(e.g., the shortest path) between the proxy and the end nodes discovered by the underlying routing protocol can be used.

The fact that nodes share keys with high probability leads to the following two observations. On one hand, it imposes threat to reveal the key nuggets because of the exposure. On the other hand, it leaves a large number of nodes to act as proxies that can secure key nuggets exchanged between end nodes. Only the compromise of the proxy will cause the associated key nugget to be revealed. Consequently, the security level of establishing a path-key will increase monotonically with the number of secured paths. Based on this observation, we propose path-key establishment scheme which leverages multiple secure paths with only one proxy for key negotiation and establishment. We propose two simple algorithms to find these proxies and compare the response time and communication overhead in terms of average number of hops and number of nodes involved to find a proxy.

If one attacker is not aiming to figure out the communication content but to cripple the system instead, he can just do so by dropping one or more of the key nuggets. To increase the robustness of the key establishment, a (k, m) threshold scheme described in [11] is adopted in our framework. In a (k, m) threshold scheme, k out of m secret shares are required to reconstruct the secret. This scheme is based on polynomial interpolation: given k points in the 2-dimensional plane $(x_1, y_1), \ldots, (x_k, y_k)$, with distinct x_i's, there is one and only one polynomial $P(x)$ of degree $k-1$ such that $y_i = P(x_i)$ for all i's. Suppose K is the path-key we randomly choose for communication. To break down K into m pieces, we randomly construct a polynomial of degree $k-1$, $P(x) = a_0 + a_1 x + \cdots + a_{k-1}x^{k-1}$, in which $a_{k-1} \neq 0$ and $K = a_0$. We then evaluate $K_1 = P(x_1), \ldots, K_i = P(x_i), \ldots, K_m = P(x_m)$. Given any subset of k pairs of these values, we can recover the coefficients of $P(x)$ by interpolation, then evaluate $K = P(0)$. However, $P(x)$ cannot be uniquely identified by less than k value pairs.

Notice that we avoid using preset x values, as in the case of the work described in [11], so that the attacker cannot gain any advantages by known-plain-text attack. For detailed description of a (k, m) threshold scheme, readers are referred to [11].

The following assumptions are made in our scheme and security analysis.

(i) Sensor nodes are not tamper resistant. Consequently, if a node is captured, the content in its memory is revealed to the attacker.
(ii) An attacker can randomly compromise at most x out of n nodes.
(iii) A routing structure has been established by a routing protocol.
(iv) Attacker cannot get keys through traffic analysis.

The notations used throughout the paper are listed in Table 1.

Table 1: Notation.

P/R	A random key predistribution scheme with key pool size P and key ring size R
K	A path-key to be established
n	Total number of nodes in the network
x	The maximum number of nodes an attacker can capture
m	Number of secure paths to set up the path-key
k	Number of secret share to recover the path-key
p	The probability that two nodes share keys
uv	Two nodes seeking private communication with each other

3.1. End-to-end key establishment scheme

Consider a network with a total number of n nodes, where each node has been loaded with a key ring drawn from a large key pool. Furthermore, assume that node u wants to set up a path-key with another node v to start a private communication. This can be achieved using the following steps.

(i) u sends out its key ID list to invite v to set up a path-key.
(ii) v randomly selects a polynomial $P(x)$ of degree $k-1$, $P(x) = a_0 + a_1 x + \cdots + a_{k-1}x^{k-1}$, where $a_{k-1} \neq 0$ and $K = a_0$. v constructs m key nuggets each of which contains a randomly selected value x and its corresponding value $P(x)$.
(iii) v then selects m proxies using one of the two approaches presented in the following section to transmit these m key nuggets to u.
(iv) Upon receiving k or more nuggets, node u reconstructs the key K by interpolation using the value pair $(x, P(x))$ carried by each nugget and uses it to securely communicate with v.

Notice that the proposed scheme does not depend on the algorithm used to produce a key. Consequently, any preassigned algorithm to produce a secure key can be adopted. An issue, which is not addressed in the above scheme, is how to select m proxies. We propose two simple methods to solve this issue. Notice that if u and v share a key, v can act as its own proxy.

The basic steps of first method to discover m proxies can be described as follows.

(i) v randomly selects m neighbors (or $m-1$ depending on whether or not v acts as its own proxy for one key nugget) and sends out request-for-proxy packets containing key IDs from both u and v.
(ii) Each recipient examines the ID list to see if it shares keys with both u and v.
    (a) If it does, it responds to v with key ID that is chosen to communicate with v.
    (b) If it does not, or it has received the same request from v, it forwards this request to a random neighbor other than the sender.

```
(1) Define
(2)   u, v: two end nodes to set up a path-key.
(3)   w: proxy candidate.
(4)   ID_u: key ID list for node u.
(5)   ID_v: key ID list for node v.
(6)   ID_self: key ID list for one node of itself.
(7)   R: request to set up path-key.
(8)   neighbors_x: 1-hop neighbors of any node x.
(9)   m: number of proxies need to be found.

(10) m_proxies (m): executed at node v

(11)  for i = 1 to m do
(12)    Randomly select a node in neighbors_v, send R
(13)  end for

(14)  if Receive positive ACK from node w, then
(15)    Register w as a proxy
(16)  end if

(17) Check_1 (R): executed at all nodes receiving R

(18)  if R is not seen before, then
(19)    if ID_self ⋂ ID_u is not empty, then
(20)      if ID_self ⋂ ID_v is not empty then
(21)        register itself as a proxy for node pair u
            and v
(22)        Send back positive ACK to node v
(23)        Exit the procedure
(24)      end if
(25)    end if
(26)  end if

(27)  Randomly select a neighbor other than the
      sender to forward R
```

Algorithm 1: The generation of *m_proxies*: *m* requests to discover *m* proxies.

The procedures used by node *v* and candidate node to select *m* proxies are outlined in Algorithm 3.1.

The second method is described as follows and is sketched in Algorithm 3.1.

(i) *v* creates a request packet and set its time-to-live (TTL) field to *t* before locally flooding it into the network. The value of *t* may be set to reflect the density of the node within the neighborhood. For dense networks, the value of *t* should be small while a large value of *t* may be required for sparse networks.

(ii) Nodes which receive a request packet respond with positive acknowledgment only if they share a key with *u* and a key with *v*, respectively.

(iii) Upon receiving *m* positive acknowledgments, *v* selects the sender of these acknowledgments as *m* proxies.[1]

[1] Notice that other schemes to select *m* proxies can be used to satisfy specific requirements such as power awareness, shortest paths, and so forth.

```
(1) Define
(2)   u, v: two end nodes to set up a path-key.
(3)   w: proxy candidate.
(4)   ID_u: key ID list for node u.
(5)   ID_v: key ID list for node v.
(6)   ID_self: key ID list for one node of itself.
(7)   R: request to set up path-key.
(8)   t: TTL (time to live) in each packet.
(9)   Timeout(t): timeout for t-hop communication.
(10)  k: number of proxies to be found.
(11)  c: counter variable.

(12) LocalFlood (t, m): executed at node v

(13)  Broadcast request including ID_u and ID_v to set
      up path-key with TTL = t
(14)  c ⇐ 0
(15)  While NOT timeout(t) do
(16)    if c == m, then
(17)      break
(18)    end if
(19)    if Receive positive ACK from node w, then
(20)      c ⇐ c + 1
(21)      Register w as a proxy
(22)    end if
(23)  end while
(24)  if c! = m, then
(25)    Increase t
(26)    LocalFlood(t, m) {incrementally flood the
        local network}
(27)  end if

(28) Check_2 (R): executed at all nodes receiving R

(29)  if R is not seen before, then
(30)    if ID_self ⋂ ID_u, is not empty then
(31)      if ID_self ⋂ ID_v is not empty, then
(32)        register itself as a proxy for node pair u
            and v
(33)        Send back positive ACK to node v
(34)      end if
(35)    end if
(36)  end if
(37)  if TTL! = 0, then
(38)    Reduce TTL
(39)    broadcast R
(40)  end if
```

Algorithm 2: *LocalFlood*: incrementally discover *k* proxies by local flooding.

Based on Algorithm 3.1, node *v* selects *m* neighbors and sends each one of them a proxy request packet. Nodes can be repeatedly selected if *v* has less than *m* direct neighbors. Notice that a request copy ceases to travel when received by a proxy. Consequently, at most *m* copies of the original requests exist in the network at any time. However, depending on the key distribution, a request may incur a large delay before discovering a proxy. If the probability of two nodes sharing keys is *p*, then the probability that a node shares key with

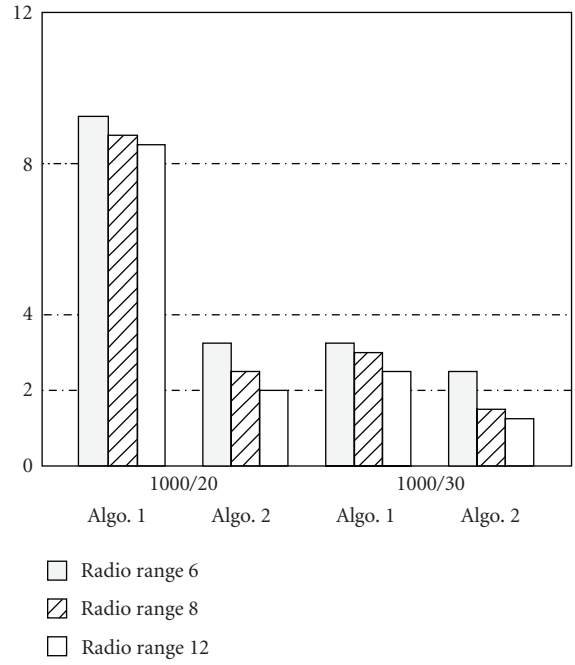

Figure 2: Average number of hops to find a proxy.

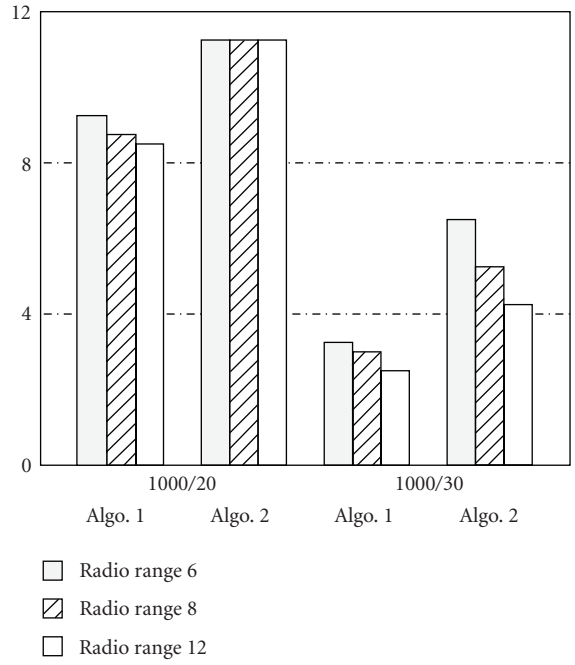

Figure 3: Average number of nodes involved to find a proxy.

two other nodes is p^2. On average, one request will need to travel $1/p^2$ nodes on average to find a proxy.

Algorithm 3.1 discovers proxies faster than Algorithm 3.1. This is specially true in dense WSNs. This algorithm, however, involves more nodes than Algorithm 3.1 because of the local flooding.

A simulation experiment was set up to compare the performance of these two algorithms. In this experiment, 1000 nodes are randomly distributed over a 100×100 square area. The radio range was varied to be 6, 8, and 12 to generate networks with different densities. In this experiment, a path-key is fragmented into 5 nuggets, therefore, 5 proxies are necessary to communicate the path-key between two end nodes. One hundred pairs of end nodes are randomly selected for 1000/20 and 1000/30 choices of pool size and ring size. Figure 2 shows the average number of hops to find a proxy. Figure 3 depicts the average number of nodes involved in discovering one proxy.

The result shows if p is large, the first approach is preferred, while the second approach should be used if the network is dense. It is therefore important that the choice of parameter p and the proxy selection method be carefully decided prior to deployment. Alternatively, a node can dynamically adapt its proxy selection strategy to use the appropriate method based on current characteristics of the network.

In further considering the simulation result, it must be pointed out that the proxies discovered by the first approach may not be physically located many hops away from node v as Figure 2 leads to believe. These proxies may actually reside within the vicinity of node v, but have been discovered through a lateral path connecting neighboring nodes located within a small number of physical hops away from node v. In fact, based on the simulation results, the sets of proxies discovered by these two approaches exhibit a large overlap.

Using either approach, only local nodes are involved in path-key establishment. Consequently the performance of the proposed scheme is independent of the network size. It only depends on the key predistribution as such a scheme scales to large-size networks.

Using the (k, m) threshold scheme, only k out of m key nuggets are required to obtain the path-key. This can ensure defending, with high probability, against random-dropping attacks staged from a captured node. However, risk still exists if the captured node sits on the crossing point of several paths between m proxies and nodes u and v and drops all the key nuggets passing through it so that not enough key nuggets can be obtained to reconstruct the path-key. Therefore, node-disjoint paths are preferred whenever possible. Techniques to find node-disjoint paths can be found in [12, 13]. Note that since all the m proxies are well dispersed around node v, the cost of finding node-disjoint paths connecting u to v can be significantly smaller in comparison to finding m node-disjoint paths directly between these two nodes.

3.2. Security analysis

The security analysis of our scheme focuses on two aspects, namely secrecy or privacy of the system and security against node capture. With respect to secrecy, the scheme must not allow any node other than the end nodes to know the shared path-key. The second aspect focuses on the likelihood that an attacker who captures a certain number of nodes may be able to obtain the key.

To evaluate the secrecy of the system, we determine the probability that x collusive nodes may cover the keys from at least k proxies used to encrypt nuggets during the path-key establishment phase, thereby violating the end nodes' exclusive path-key sharing property. The vulnerability of the system to node capture is measured by computing the likelihood that an attacker who captures x nodes may obtain at least k key nuggets.

For simplicity, we assume that there are $2m$ distinct keys used to secure key nuggets by m proxies. In this case, colluding nodes will need to have both keys used by one proxy to be sure that they can correctly obtain the key nugget being transported by that proxy. Consider a set of x collusive nodes. The probability, P_r, for one of the $2m$ keys to be installed onto a given network node is P_r = key ring size/key pool size = R/P. Then the probability of this key being contained in the union of the x colluding nodes is $1-(1-P_r)^x$. Therefore, the probability, P_x, that colluding x nodes cover at least k pairs of keys used to secure the key nuggets is

$$P_x = \sum_{l=k}^{m} \left(1 - \left(1 - \frac{R}{P}\right)^x\right)^{2l}. \tag{1}$$

Note this probability is independent of the number of nodes deployed. As such, this probability defines the system security for a given key pool size and key ring size. Furthermore, these m proxies may use less than m pairs of keys to secure m key nuggets. Consequently, (1) gives a lower bound of the probability that a set of x collusive nodes cover at least k pairs of the keys used to securely set up the path-key.

Figure 4 depicts the probability of x collusive nodes covering at least k pairs of keys. We observe that as x increases, P_x increases rapidly. It is desired that we keep P_r small, however, this will affect the choice of method to discover the proxies. It is left to the designer to choose appropriate network specification given a required security level. Furthermore, we can use the cooperation scheme in [7], whereby a proxy uses all the keys that it shares with one end node to encrypt the key nugget to further reduce the likelihood of at least k pairs of keys being covered.

Another observation, which can be made based on the results of Figure 4, is that even when $k = 3$ and $m = 5$, a set of 50 colluding nodes is required to recover at least k key nuggets with probability of 45.4%. It is therefore unlikely that a smaller number of colluding nodes can determine the path-key by overhearing the traffic.

The vulnerability of the network to node capture depends on the ability of the attacker to acquire the key nuggets directly. If either u or v is captured, the path-key is revealed. In a network of n nodes, if x ($x > m$) nodes are captured, the probability, P_1, that one or both end nodes are among these nodes is

$$P_1 = \frac{\binom{2}{1}\binom{n-2}{x-1} + \binom{2}{2}\binom{n-2}{x-2}}{\binom{n}{x}} = \frac{(2n-x-1)/(x-1) \times \binom{n-2}{x-2}}{\binom{n}{x}}. \tag{2}$$

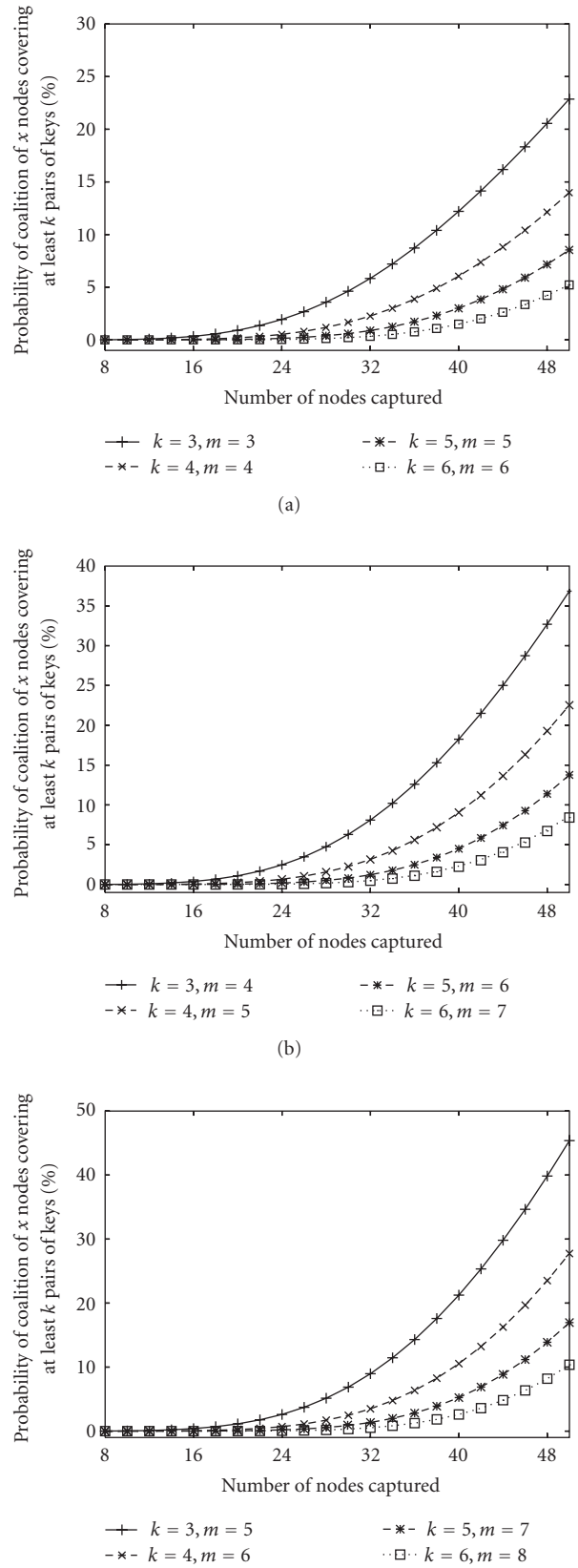

Figure 4: Probability of x collusive nodes covering at least k pairs of keys. Zero redundancy in (a), 1-packet redundancy in (b), and 2-packet redundancy in (c).

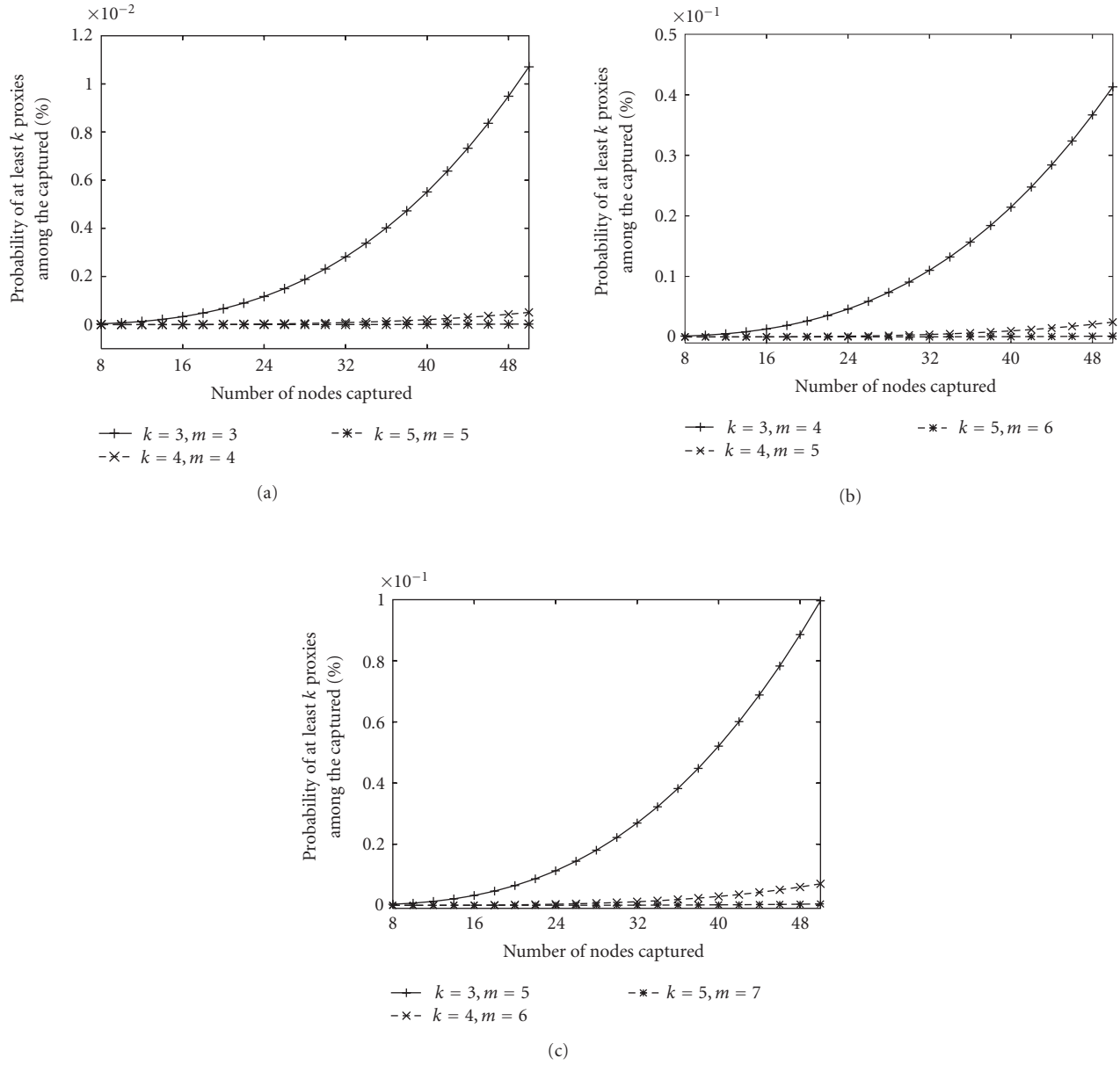

Figure 5: Probability of at least k proxies are among x nodes being captured. Zero redundancy in (a), 1-packet redundancy in (b), and 2-packet redundancy in (c).

The probability, P_2, that x nodes contain no end nodes but cover at least k proxies is

$$P_2 = (1-p) \times \sum_{l=k}^{m} \frac{\binom{m}{l}\binom{n-m-2}{x-l}}{\binom{n}{x}}, \tag{3}$$

where p is the probability that two nodes share keys. The factor $(1-p)$ in (3) accounts for the fact that m proxies are used. Hence, the probability P_c of all keys shared being revealed after the capture of x nodes is

$$\begin{aligned} P_c &= P_1 + P_2 \\ &= \frac{((2n-x-1)/(x-1))\binom{n-2}{x-2} + (1-p) \times \sum_{l=k}^{m} \binom{m}{l}\binom{n-m-2}{x-l}}{\binom{n}{x}}. \end{aligned} \tag{4}$$

We can see that the first term in (4) is solely dependent on the scale of the network deployment and the number of nodes captured. No scheme can protect the capture of communicating end nodes unless the nodes are tamper-proof. We are therefore more interested in the second term which

relates the number of paths used and the scale of the system. It can be noted that the factor $(1 - p)$ is omitted because once the key pool size and key ring size are chosen, this factor becomes a constant. The plot describing the variation of $P = \sum_{l=k}^{m}(\binom{m}{l}\binom{n-m-2}{x-l}/\binom{n}{x})$ as a function of x is depicted in Figure 5, for $n = 1000$ and various values of k and m. P is the probability that at least k proxies are among those x captured nodes.

Based on the result, a satisfactory security level (7×10^{-3}%) can be achieved even when a large percentage of nodes (5%) are captured and k is small with endurance of 2 packets loss ($k = 4$, $m = 6$). Furthermore, there is a noticeable jump from $k = 3$ to $k = 4$ in all cases which suggests that we invest a little more using 4 instead of 3 proxies whenever possible to achieve a big security improvement.

There are obviously trade-offs among the choices of values for k and m. Larger m incurs more overhead but leaves more room for robustness consideration. Larger k means more security yet less robustness. In the extreme case when $m = k$, there is *zero* tolerance for lost packets. Users should pick up values for k and m according to the security, energy budget, and robustness requirements of their applications.

4. CONCLUSION

This paper addresses the path-key establishment exposure problem commonly encountered in key predistribution schemes in WSNs. We propose a robust path-key establishment framework, which uses multiple secured paths for the negotiation and exchange of symmetric keys between end nodes. Since the scheme ensures that each key share can be revealed only to one node on each path, the exposure of that key nugget is minimized. The analysis shows that the proposed scheme can greatly improve the security of key establishment. Furthermore this scheme assumes no specific routing protocols, and therefore, it is not dependent on the physical topology of the network. As long as the network is connected and there are enough nodes deployed, the proposed scheme can be incorporated to most key predistribution schemes without significant changes. Robustness is achieved through redundant information such that not all packets are required to obtain the key.

REFERENCES

[1] D. W. Carman, P. S. Kruus, and B. J. Matt, "Constrains and approaches for distributed sensor netowrk security," Tech. Rep. 00-010, NAI Labs, Glenwood, Md, USA, September 2000.

[2] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS '02)*, pp. 41–47, Washington, DC, USA, November 2002.

[3] H. Ling and T. Znati, "End-to-end pairwise key establishment using multi-path in wireless sensor network," in *Proceedings of IEEE Global Communications Conference (GLOBECOM '05)*, St. Louis, Mo, USA, November-December 2005.

[4] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of IEEE Symposium on Security and Privacy (S&P '03)*, pp. 197–213, Berkeley, Calif, USA, May 2003.

[5] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge," in *Proceedings of 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, vol. 1, pp. 586–597, Hong Kong, March 2004.

[6] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 42–51, Washingtion, DC, USA, October 2003.

[7] R. D. Pietro, L. V. Mancini, and A. Mei, "Random key-assignment for secure wireless sensor networks," in *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '03)*, pp. 62–71, Fairfax, Va, USA, October 2003.

[8] D. Liu and P. Ning, "Improving key predistribution with deployment knowledge in static sensor networks," *ACM Transactions on Sensor Networks*, vol. 1, no. 2, pp. 204–239, 2005.

[9] D. Liu, P. Ning, and W. Du, "Group-based key pre-distribution in wireless sensor networks," in *Proceedings of ACM Workshop on Wireless Security (WiSe '05)*, Cologne, Germany, September 2005.

[10] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Establishing pairwise keys for secure communication in ad hoc networks: a probabilistic approach," in *Proceedings of 11th IEEE International Conference on Network Protocols (ICNP '03)*, pp. 326–335, Atlanta, Ga, USA, November 2003.

[11] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[12] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly resilient, energy efficient multipath routing in wireless sensor networks," *Mobile Computing and Communications Review*, vol. 1, no. 2, pp. 10–24, 2002.

[13] X. Li and L. Cuthbert, "A reliable node-disjoint multipath routing with low overhead in wireless ad hoc networks," in *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (ACM MSWiM '04)*, pp. 230–233, Venice, Italy, October 2004.

Guanfeng Li is currently a Ph.D. student at Department of Computer Science, University of Pittsburgh. He got his B.E. degree from the Department of Computer Science and Technology at Tsinghua University in 1999 and his M.S. degree from Computer Science Department at University of Central Florida in 2001. His research interest is to develop algorithms for routing, congestion control, and security in wireless ad hoc and sensor networks. He is a Student Member of IEEE since 2005.

Hui Ling is a Ph.D. student at Department of Computer Science, University of Pittsburgh. He got his B.S. and M.S. degrees from Nanjing University in 1999 and 2002, respectively. His major research interest includes routing and security protocols in mobile ad hoc and wireless sensor networks. He is currently investigating the key-predistribution protocols in wireless sensor networks. He is a Student Member of IEEE.

Taieb Znati is currently a Professor in the Department of Computer Science, with a joint appointment in telecommunications in the Department of Information Science at the University of Pittsburgh. He also served as a Senior Program Director for networking research at the National Science Foundation, in the Division of Advanced Networking Infrastructure and Research, and later in the Division of Computer and Network Systems within the Computer Information Systems and Engineering Directorate. In the fourth year of his tenure at NSF, he served as the Chair of the Information Technology Research Initiative (ITR), an NSF cross-directorate program. Dr. Znati's current research interests focus on the design of network architectures and protocols for wired and wireless communication networks to support applications' QoS and security requirements. Dr. Znati served as the General Chair of IEEE INFOCOM 2005, SECON 2004, the first IEEE conference on sensor and ad hoc communications and networks, the annual simulation symposium, and UbiCare'06 the first workshop on ubiquitous and pervasive healthcare. He is a Member of the Editorial Board of the International Journal of Parallel and Distributed Systems and Networks, the Pervasive and Mobile Computing Journal, the Journal on Wireless Communications and Mobile Computing.

Weili Wu received her M.S. and Ph.D. degrees in computer science both from University of Minnesota, in 1998 and 2002, respectively. She is currently an Assistant Professor and a Lab Director of the Database Research Lab at the Department of Computer Science and Engineering, the University of Texas at Dallas. Her research interest is mainly in database systems, especially in spatial database with applications in geographic information systems and bioinformatics, distributed database in Internet system, and wireless database systems with connection to wireless communication. She has published more than 40 research papers in various prestigious journals and conferences such as IEEE Transaction on Multimedia, Theoretical Computer Science, Journal of Complexity, Discrete Mathematics, Discrete Applied Mathematics, ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, SIAM Conference on Data Mining, UCGIS Summer Assembly, and International Conference on Computer Science and Informatics. She is an author of the textbook *Mathematical Theory of Optimization* and an Editor of the research monograph *Clustering and Information Retrieval*. She is an Associate Editor of KAIS: An International Journal on Knowledge and Information Systems and a Member of the Editorial Board of IJBRA International Journal of Bioinformatics Research and Applications. She is a Member of the IEEE Computer Society.