Establishing Pairwise Keys in Distributed Sensor Networks

DONGGANG LIU, PENG NING and RONGFANG LI
North Carolina State University

Pairwise key establishment is a fundamental security service in sensor networks; it enables sensor nodes to communicate securely with each other using cryptographic techniques. However, due to the resource constraints on sensor nodes, it is not feasible to use traditional key management techniques such as public key cryptography and key distribution center (KDC). A number of key predistribution techniques have been proposed for pairwise key establishment in sensor networks recently. To facilitate the study of novel pairwise key predistribution techniques, this paper develops a general framework for establishing pairwise keys between sensor nodes using bivariate polynomials. This paper then proposes two efficient instantiations of the general framework: a *random subset assignment* key predistribution scheme, and a *hypercube-based* key predistribution scheme. The analysis shows that both schemes have a number of nice properties, including high probability, or guarantee to establish pairwise keys, tolerance of node captures, and low storage, communication, and computation overhead. To further reduce the computation at sensor nodes, this paper presents an optimization technique for polynomial evaluation, which is used to compute pairwise keys. This paper also reports the implementation and the performance of the proposed schemes on MICA2 motes running TinyOS, an operating system for networked sensors. The results indicate that the proposed techniques can be applied efficiently in resource constrained sensor networks.

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*; D.4.6 [**Operating Systems**]: Security and Protection—*Cryptographic controls*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms: Security, Design, Algorithms

Additional Key Words and Phrases: Sensor Networks, Key Management, Key Predistribution, Pairwise Key

1. INTRODUCTION

Distributed sensor networks have received a lot of attention recently due to its wide applications in military as well as civilian operations. Example applications include target

This work is partially supported by the National Science Foundation (NSF) under grant CNS-0430223. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of NSF, or the U.S. Government.

A preliminary version this paper appeared in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pages 52–61, October 2003.

Authors' address: Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8207; emails: dliu@ncsu.edu, pning@ncsu.edu, and rongfangli@hotmail.com.

tracking, scientific exploration, and data acquisition in hazardous environments. The sensor nodes are typically small, low-cost, battery powered, and highly resource constrained. They usually communicate with each other through wireless links.

Security services such as authentication and key management are critical to secure the communication between sensor nodes in hostile environments. As one of the most fundamental security services, pairwise key establishment enables the sensor nodes to communicate securely with each other using cryptographic techniques. However, due to the resource constraints on sensor nodes, it is not feasible for them to use traditional pairwise key establishment techniques such as public key cryptography and key distribution center (KDC).

Instead of the above two techniques, sensor nodes may establish keys between each other through *key predistribution*, where keying materials are predistributed to sensor nodes before deployment. As two extreme cases, one may setup a *global* key among the network so that two sensor nodes can establish a key based on this global key, or assign each sensor node a unique random key with each of the other nodes. However, the former is vulnerable to the compromise of a single node, and the latter introduces huge storage overhead on sensor nodes.

Eschenauer and Gligor proposed a probabilistic key predistribution scheme recently for pairwise key establishment [Eschenauer and Gligor 2002]. The main idea is to let each sensor node randomly pick a set of keys from a key pool before the deployment so that any two sensor nodes have a certain probability to share at least one common key. Chan et al. further extended this idea and developed two key predistribution techniques: a q-composite key predistribution scheme and a random pairwise keys scheme [Chan et al. 2003]. The q-composite key predistribution also uses a key pool but requires two nodes compute a pairwise key from at least q predistributed keys that they share. The random pairwise keys scheme randomly picks pairs of sensor nodes and assigns each pair a unique random key. Both schemes improve the security over the basic probabilistic key predistribution scheme.

However, the pairwise key establishment problem is still not fully solved. For the basic probabilistic and the q-composite key predistribution schemes, as the number of compromised nodes increases, the fraction of affected pairwise keys increases quickly. As a result, a small number of compromised nodes may affect a large fraction of pairwise keys. Though the random pairwise keys scheme does not suffer from the above security problem, given a memory constraint, the network size is strictly limited by the desired probability that two sensor nodes share a pairwise key, the memory available for keys on sensor nodes, and the number of neighbor nodes that a sensor node can communicate with.

In this paper, we develop a number of key predistribution techniques to deal with the above problems. We first develop a general framework for pairwise key establishment based on the polynomial-based key predistribution protocol in [Blundo et al. 1993] and the probabilistic key distribution in [Eschenauer and Gligor 2002; Chan et al. 2003]. This framework is called *polynomial pool-based key predistribution*, which uses a polynomial pool instead of a key pool in [Eschenauer and Gligor 2002; Chan et al. 2003]. The secrets on each sensor node are generated from a subset of polynomials in the pool. If two sensor nodes have the secrets generated from the same polynomial, they can establish a pairwise key based on the polynomial-based key predistribution scheme. All the previous schemes in [Blundo et al. 1993; Eschenauer and Gligor 2002; Chan et al. 2003] can be considered as special instances in this framework.

By instantiating the components in this framework, we further develop two novel pairwise key predistribution schemes: a *random subset assignment* scheme and a *hypercube-based* scheme. The random subset assignment scheme assigns each sensor node the secrets generated from a random subset of polynomials in the polynomial pool. The hypercube-based scheme arranges polynomials in a hypercube space, assigns each sensor node to a unique coordinate in the space, and gives the node the secrets generated from the polynomials related to the corresponding coordinate. Based on this hypercube, each sensor node can then identify whether it can directly establish a pairwise key with another node, and if not, what intermediate nodes it can contact to indirectly establish the pairwise key.

Our analysis indicates that our new schemes have some nice features compared with the previous methods. In particular, when the fraction of compromised secure links is less than 60%, given the same storage constraint, the random subset assignment scheme provides a significantly higher probability of establishing secure communication between non-compromised nodes than the previous methods. Moreover, unless the number of compromised nodes sharing a common polynomial exceeds a threshold, compromise of sensor nodes does not lead to the disclosure of keys established between non-compromised nodes using this polynomial.

Similarly, the hypercube-based scheme also has a number of attractive properties. First, it guarantees that any two nodes can establish a pairwise key when there are no compromised nodes, provided that the sensor nodes can communicate with each other. Second, it is resilient to node compromise. Even if some sensor nodes are compromised, there is still a high probability to re-establish a pairwise key between non-compromised nodes. Third, a sensor node can directly determine whether it can establish a pairwise key with another node and how to compute the pairwise key if it can. As a result, there is no communication overhead during the discovery of directly shared keys.

Evaluation of polynomials is essential to the proposed schemes, since it affects the performance of computing a pairwise key. To reduce the computation at sensor nodes, we provide an optimization technique for polynomial evaluation. The basic idea is to compute multiple pieces of key fragments over some special finite fields such as F_{2^8+1} and $F_{2^{16}+1}$, and concatenate these fragments into a regular key. A nice property provided by such finite fields is that no division is necessary for modular multiplication. As a result, evaluation of polynomials can be performed efficiently on low cost processors on sensor nodes that do not have division instructions. Our analysis indicates that such a method only slightly decreases the uncertainty of the keys.

We have implemented the aforementioned algorithm on MICA2 motes [Crossbow Technology Inc.] running TinyOS [Hill et al. 2000]. The implementation only occupies a small amount of memory (e.g. 416 bytes in ROM and 20 bytes in RAM for one of our implementations excluding the memory for polynomial coefficients). The evaluation indicates that computing a 64-bit key using this technique can be faster than generating a 64-bit MAC (Message Authentication Code) using RC5 [Rivest 1994] or SkipJack [NIST 1998] for a reasonable degree of polynomial. These results show that our schemes are practical for resource constrained sensor networks.

The rest of this paper is organized as follows. Section 2 gives an overview of the polynomial-based key predistribution technique. Section 3 presents our general framework for polynomial pool-based key predistribution. Sections 4 and 5 describe the random subset assignment scheme and the hypercube-based scheme, respectively. Section 6 presents the technique to reduce the computation at sensor nodes, and reports our implementation and performance results. Section 7 discusses the related work. Section 8 concludes this paper and points out some future research directions.

2. POLYNOMIAL-BASED KEY PREDISTRIBUTION FOR SENSOR NETWORKS

In this section, we briefly review the basic polynomial-based key predistribution protocol in [Blundo et al. 1993], which is the basis of our new techniques. The protocol in [Blundo et al. 1993] was developed for group key predistribution. Since our goal is to establish pairwise keys, for simplicity, we only discuss the special case of pairwise key establishment in the context of sensor networks.

To predistribute pairwise keys, the (key) setup server randomly generates a bivariate t-degree polynomial $f(x, y) = \sum_{i,j=0}^{t} a_{ij}x^i y^j$ over a finite field F_q, where q is a prime number that is large enough to accommodate a cryptographic key, such that it has the property of $f(x, y) = f(y, x)$. (In the following, we assume all the bivariate polynomials have this property without explicit statement.) It is assumed that each sensor node has a unique ID. For each node i, the setup server computes a *polynomial share* of $f(x, y)$, that is, $f(i, y)$. This polynomial share is predistributed to node i. Thus, for any two sensor nodes i and j, node i can compute the key $f(i, j)$ by evaluating $f(i, y)$ at point j, and node j can compute the same key $f(j, i) = f(i, j)$ by evaluating $f(j, y)$ at point i. As a result, nodes i and j can establish a common key $f(i, j)$.

In this approach, each sensor node i needs to store a t-degree polynomial $f(i, x)$, which occupies $(t + 1) \log q$ storage space. To establish a pairwise key, both sensor nodes need to evaluate the polynomial at the ID of the other sensor node. (In Section 6, we will present techniques to reduce the computation required to evaluate polynomials.) There is no communication overhead during the pairwise key establishment process.

The security proof in [Blundo et al. 1993] ensures that this scheme is unconditionally secure and t-collusion resistant. That is, the coalition of no more than t compromised sensor nodes knows nothing about the pairwise key between any two non-compromised nodes.

It is theoretically possible to use the general group key distribution protocol in [Blundo et al. 1993] in sensor networks. However, the storage cost for a polynomial share is exponential in terms of the group size, making it prohibitive in sensor networks. In this paper, we will focus on the problem of pairwise key establishment.

3. POLYNOMIAL POOL-BASED KEY PREDISTRIBUTION

The polynomial-based key predistribution scheme discussed in Section 2 has some limitations. In particular, it can only tolerate the collusion of no more than t compromised nodes, where the value of t is limited by the available memory space and the computation capability on sensor nodes. Indeed, the larger a sensor network is, the more likely an adversary compromises more than t sensor nodes and then the entire network.

To have secure and practical key establishment techniques, we develop a general framework for key predistribution based on the scheme presented in Section 2. We call it the *polynomial pool-based key predistribution*, since a pool of random bivariate polynomials are used in this framework. In this section, we focus on the discussion of this general framework. In the next two sections, we will present two efficient instantiations of this framework.

The polynomial pool-based key predistribution is inspired by the studies in [Eschenauer and Gligor 2002; Chan et al. 2003]. The basic idea can be considered as the combination of the polynomial-based key predistribution and the key pool idea used in [Eschenauer and Gligor 2002; Chan et al. 2003]. However, our framework is more general in that it allows different choices to be instantiated within this framework, including those in [Eschenauer and Gligor 2002; Chan et al. 2003] and our later instantiations in Sections 4 and 5.

Intuitively, this general framework generates a pool of random bivariate polynomials and assigns shares on a subset of bivariate polynomials in the pool to each sensor node. The polynomial pool has two special cases. When it has only one polynomial, the general framework degenerates into the polynomial-based key predistribution. When all the polynomials are 0-degree ones, the polynomial pool degenerates into a key pool used in [Eschenauer and Gligor 2002; Chan et al. 2003].

Pairwise key establishment in this framework has three phases: *setup*, *direct key establishment*, and *path key establishment*. The setup phase is performed to initialize the nodes by distributing polynomial shares to them. After being deployed, if two sensor nodes need to establish a pairwise key, they first attempt to do so through direct key establishment. If they can successfully establish a common key, there is no need to start path key establishment; otherwise, these two nodes start path key establishment, trying to establish a pairwise key with the help of other sensor nodes.

3.1 Phase 1: Setup

The setup server randomly generates a set $\mathcal{F}$ of bivariate t-degree polynomials over the finite field F_q. To identify different polynomials, the setup server may assign each polynomial a unique ID. For each sensor node i, the setup server picks a subset of polynomials $\mathcal{F}_i \subseteq \mathcal{F}$, and assigns the shares of these polynomials to node i. The main issue in this phase is the *subset assignment* problem, which specifies how to pick a subset of polynomials from $\mathcal{F}$ for each sensor node.

Here we identify two ways to perform subset assignments: *random assignment* and *predetermined assignment*.

3.1.1 *Random Assignment.*

With random assignment, the setup server randomly picks a subset of $\mathcal{F}$ for each sensor node. This random selection should be evenly distributed in $\mathcal{F}$ for security concerns; otherwise, some polynomials may have higher probability of being selected and higher frequency of being used in key establishment than the others and thus become the primary targets of attacks. Several parameters may be used to control this process, including the number of polynomial shares assigned to a node and the size of $\mathcal{F}$. In the simplest case, the setup server assigns the same number of random selected polynomial shares to each sensor node.

3.1.2 *Predetermined Assignment.*

When predetermined assignment is used, the setup server follows certain scheme to assign subsets of $\mathcal{F}$ to sensor nodes. A predetermined assignment should bring some nice properties that can be used to improve direct and path key establishment.

3.2 Phase 2: Direct Key Establishment

A sensor node starts phase 2 if it needs to establish a pairwise key with another node. If both sensor nodes have shares on the same bivariate polynomial, they can establish the pairwise key directly using the polynomial-based key predistribution discussed in Section

2. The main issue in this phase is the *polynomial share discovery* problem, which specifies how to find a common bivariate polynomial of which both nodes have polynomial shares. For convenience, we say two sensor nodes have a *secure link* if they can establish a pairwise key through direct key establishment. A pairwise key established in this phase is called a *direct key*.

Here we identify two types of techniques to solve this problem: *predistribution* and *real-time discovery*.

3.2.1 *Predistribution.* The setup server predistributes certain information to the sensor nodes, so that given the ID of another sensor node, a sensor node can determine whether it can establish a direct key with the other node. A naive method is to let each sensor node store the IDs of all the sensor nodes with which it can setup direct keys. However, this naive method has difficulties in dealing with the sensor nodes that join the network on the fly, because the setup server has to inform some existing nodes about the addition of new sensor nodes. Alternatively, the setup server may map the ID of each sensor node to the IDs of polynomial shares it has so that given the ID of a sensor node, anybody can derive the IDs of polynomial shares it has. Thus, any sensor node can determine immediately whether it can establish a direct key with a given sensor node by only knowing its ID. Note that this method requires the predetermined assignment strategy in the setup phase.

The drawback of predistribution methods is that an attacker may also know the distribution of the polynomial shares. As a result, the attacker may precisely target at certain sensor nodes, attempting to learn the shares of a particular bivariate polynomial. The following alternative way may avoid this problem.

3.2.2 *Real-time discovery.* Intuitively, real-time discovery requires two sensor nodes to discover on the fly whether they have shares on a common bivariate polynomial. As one possible way, two nodes may first exchange the IDs of polynomials of which they both have shares, and then try to identify the common polynomial. To protect the IDs of the polynomials, the sensor node may challenge the other party to solve puzzles instead of disclosing the IDs of the polynomials directly. Similar to the the method in [Eschenauer and Gligor 2002], when node i needs to establish a pairwise key with node j, it sends node j an encryption list, $\alpha, E_{K_v}(\alpha), v = 1, ..., |\mathcal{F}_i|$, where K_v is computed by evaluating the v^{th} polynomial share in $\mathcal{F}_i$ on point j (a potential pairwise key node j may have). When node j receives this encryption list, it first computes $\{K'_v\}_{v=1,...,|\mathcal{F}_j|}$, where K'_v is computed by evaluating the v^{th} polynomial share in $\mathcal{F}_j$ on point i (a potential pairwise key node i may have). Node j then generates another encryption list $\{E_{K'_v}(\alpha)\}_{v=1,...,|\mathcal{F}_j|}$. If there exists a common encryption value that is included in both encryption lists, node i and node j can establish a common key, which is the key used to generate this common value.

The drawback of real-time discovery is that it introduces additional communication overhead that does not appear in the predistribution approaches. If the polynomial IDs are exchanged in clear text, an attacker may gradually learn the distribution of polynomials among sensor nodes and selectively capture and compromise sensor nodes based on this information. However, it is more difficult for an adversary to collect the polynomial distribution information in the real-time discovery method than in the predistribution method, since the adversary has to monitor the communication among sensor nodes. In addition, when the encryption list is used to protect the IDs of polynomial shares in a sensor nodes, an adversary has no way to learn the polynomial distribution among sensor nodes and thus cannot launch selective node capture attacks.

3.3 Phase 3: Path Key Establishment

If direct key establishment fails, two sensor nodes need to start phase 3 to establish a pairwise key with the help of other sensor nodes. To establish a pairwise key with node j, a sensor node i needs to find a sequence of nodes between itself and node j such that any two adjacent nodes in this sequence can establish a direct key. For the sake of presentation, we call such a sequence of nodes a *key path* (or simply a *path*), since the purpose of such a path is to establish a pairwise key. Then either node i or j initiates a key establishment request with the other node through the intermediate nodes along the path. A pairwise key established in this phase is called an *indirect key*. A subtle issue is that two adjacent nodes in the path may not be able to communicate with each other directly. In this paper, we assume that they can always discover a route between themselves so that the messages from one node can reach the other.

The main issue in this phase is the *path discovery* problem, which specifies how to find a path between two sensor nodes. Similar to phase 2, there are two types of techniques to address this problem.

3.3.1 *Predistribution.* Similar to the predistribution technique in phase 2, the setup server predistributes certain information to each sensor node so that given the ID of another node, each node can determine at least one key path to the other node directly. The resulting key path is called the *predetermined key path*. For convenience, we call the process to compute the predetermined key paths *key path predetermination*. The drawback is that an attacker may also take advantage of the predistributed information to attack the network. Moreover, it is possible that none of the predetermined key paths is available to establish an indirect key between two nodes due to compromised (intermediate) nodes or communication failures.

To deal with the above problem, the source node needs to dynamically find other key paths to establish an indirect key with the destination node. For convenience, we call such a process *dynamic key path discovery*. For example, the source node may contact a number of other nodes with which it can establish direct keys using non-compromised polynomials, attempting to find a node that has a path to the destination node to help establish an indirect key.

3.3.2 *Real-time discovery.* Real-time discovery techniques have the sensor nodes discover key path on the fly without any predetermined information. The sensor nodes may take advantage of the direct keys established through direct key establishment. For example, to discover a key path to another sensor node, a sensor node picks a set of intermediate nodes with which it has established direct keys. The source node may send request to all these intermediate nodes. If one of the intermediate nodes can establish a direct key with the destination node, a key path is discovered. Otherwise, this process may continue with the intermediate nodes forwarding the request. Such a process is similar to a route discovery process used to establish a route between two nodes. The drawback is that such methods may introduce substantial communication overhead.

4. KEY PREDISTRIBUTION USING RANDOM SUBSET ASSIGNMENT

In this section, we present the first instantiation of the general framework by using a random strategy for the subset assignment during the setup phase. That is, for each sensor node, the setup server selects a random subset of polynomials in $\mathcal{F}$ and assigns their polynomial shares to the node.

4.1 The Random Subset Assignment Scheme

The random subset assignment scheme can be considered as an extension to the basic probabilistic scheme in [Eschenauer and Gligor 2002]. Instead of randomly selecting keys from a large key pool and assigning them to sensor nodes, our method randomly chooses polynomials from a polynomial pool and assigns their polynomial shares to sensor nodes. However, our scheme also differs from the scheme in [Eschenauer and Gligor 2002]. In [Eschenauer and Gligor 2002], the same key may be shared by multiple sensor nodes. In contrast, in our scheme, there is a unique key for each pair of sensor nodes. If no more than t shares on the same polynomial are disclosed, none of the pairwise keys constructed using this polynomial between two non-compromised sensor nodes will be disclosed.

Now let us describe this scheme by instantiating the three components in the general framework.

(1) **Subset assignment:** The setup server randomly generates a set $\mathcal{F}$ of s bivariate t-degree polynomials over the finite field F_q. For each sensor node, the setup server randomly picks a subset of s' polynomials from $\mathcal{F}$ and assigns shares as well as the IDs of these s' polynomials to the sensor node.

(2) **Polynomial share discovery:** Since the setup server does not predistribute enough information to the sensor nodes for polynomial share discovery, sensor nodes that need to establish a pairwise key have to find out a common polynomial with real-time discovery techniques. To discover a common bivariate polynomial, the source node discloses a list of polynomial IDs to the destination node. If the destination node finds that they have shares on the same polynomial, it informs the source node the ID of this polynomial; otherwise, it replies with a message that contains a list of its polynomial IDs, which also indicates that the direct key establishment fails.

(3) **Path discovery:** If two sensor nodes fail to establish a direct key, they need to start path key establishment phase. During this phase, the source node tries to find another node that can help it setup a pairwise key with the destination node. Basically, the source node broadcasts two list of polynomial IDs. One includes the polynomial IDs at the source node, and the other includes the polynomial IDs at the destination node. These two lists are available at both the source and the destination nodes after the polynomial share discovery. If one of the nodes that receives this request is able to establish direct keys with both the source and the destination nodes, it replies with a message that contains two encrypted copies of a randomly generated key: one encrypted by the direct key with the source node, and the other encrypted by the direct key with the destination node. Both the source and the destination nodes can then get the new pairwise key from this message. (Note that the intermediate node acts as an ad hoc KDC in this case.) In practice, we may restrict that a sensor node only contact its neighbors within a certain range.

4.2 Performance

Similar to the analysis in [Eschenauer and Gligor 2002], the probability of two sensor nodes sharing the same bivariate polynomial, which is the probability that two sensor nodes can establish a direct key, can be estimated by

$$p = 1 - \prod_{i=0}^{s'-1} \frac{s - s' - i}{s - i}. \tag{1}$$

Figure 1(a) shows the relationship between p and the combinations of s and s'. It is easy to see that the closer s and s' are, the more likely two sensor nodes can establish a direct key. Our later security analysis (in Section 4.4) shows that small s and s' can provide high security performance. This differs from the the basic probabilistic scheme [Eschenauer and Gligor 2002] and the q-composite scheme [Chan et al. 2003], where the key pool size has to be very large to meet certain security requirement. The reason is that there is another parameter (i.e., the degree t of the polynomials) that affects the security performance of the random subset assignment scheme. In Equation 1, the value of s' is affected by the storage overhead and the degree t of the polynomials. In fact, we have $t = \frac{C}{s'} - 1$, where C is the number keys a sensor node can store.

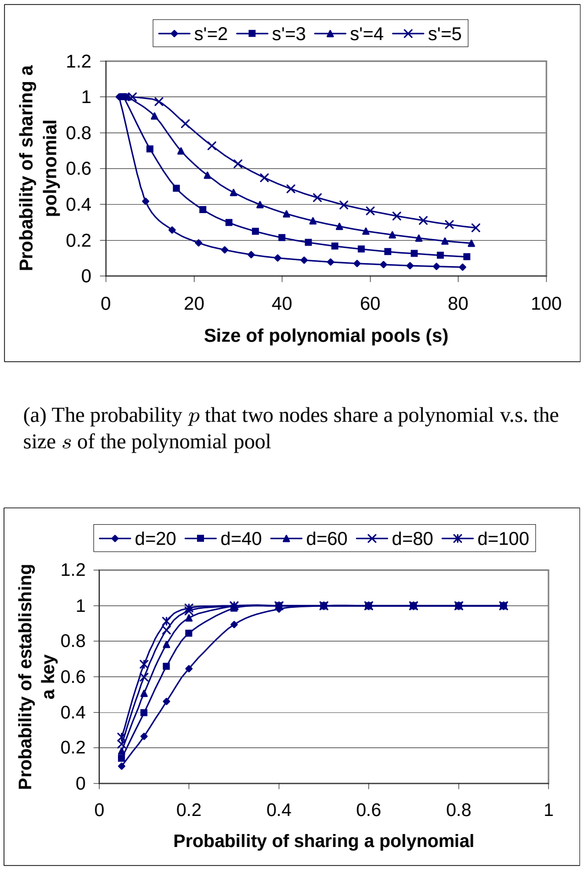
Now let us consider the probability that two nodes can establish a key through either the polynomial share discovery or the path discovery. Let d denote the average number of neighbor nodes that each sensor node contacts. Consider any one of these d nodes. The probability that it shares direct keys with both the source and the destination nodes is p^2, where p is computed by Equation 1. As long as one of these d nodes can act as an intermediate node, the source and the destination nodes can establish a common key. It follows that the probability of two nodes establishing a pairwise key (directly or indirectly) is $P_s = 1-(1-p)(1-p^2)^d$. Figure 1(b) shows that the probability P_s of establishing a pairwise key between two sensor nodes increases quickly as the probability p of establishing direct keys or the number d of neighbor nodes it contacts increases.

4.3 Overheads

Each node has to store s' t-degree polynomials over F_q, which introduces $s'(t+1)\log q$ bits storage overhead. In addition, each node needs to remember the IDs of revoked nodes with which it can establish direct keys. Assume the IDs of sensor nodes are chosen from a finite field $F_{q'}$. The storage overhead introduced by the revoked IDs is at most $s't\log q'$ bits, since if $t+1$ shares of one bivariate polynomial are revoked, this polynomial is compromised and can be discarded. Thus, the overall storage overhead is at most $s'(t+1)\log q + s't\log q'$ bits.

In terms of communication overhead, during the polynomial share discovery, the source node needs to disclose a list of s' IDs to the destination node. The communication overhead is mainly due to the transmission of such lists. During the path discovery, the source node broadcasts a request message that consists of two lists of polynomial IDs. This introduces one broadcast message at the source node and possibly several broadcast messages at other nodes receiving this request if they further forward this request. However, due to the small values of s and s' in our scheme, all the broadcast messages are small messages, and can be done efficiently in resource constrained sensor networks. If a node receiving this message shares common polynomials with both the source and the destination nodes, it only needs to reply with a message consisting of two encrypted copies of a randomly generated key.

In terms of computational overhead, the polynomial share discovery requires one polynomial evaluation at each node if they share a common polynomial. During the path discovery, if a node receiving the request shares common polynomials with both the source



(a) The probability p that two nodes share a polynomial v.s. the size s of the polynomial pool

(b) The probability P_s of establishing a pairwise key v.s. the probability p that two nodes share a polynomial

Fig. 1. Probabilities about pairwise key establishment

and the destination nodes, it only needs to perform two polynomial evaluations and two encryptions. If there exists at least one intermediate node that can be used in the establishment of an indirect key, both the source and the destination nodes only need one decryption.

4.4 Security Analysis

It follows from the security analysis in [Blundo et al. 1993] that an attacker cannot determine non-compromised keys established with a polynomial if he/she has compromised no more than t sensor nodes that have shares of this polynomial. Assume an attacker randomly compromises N_c sensor nodes, where $N_c > t$. Consider any polynomial f in $\mathcal{F}$. The probability of f being chosen for a sensor node is $\frac{s'}{s}$, and the probability of this polynomial being chosen exactly i times among N_c compromised sensor nodes is

$$P[i \text{ compromised shares}] = \frac{N_c!}{(N_c - i)!i!}(\frac{s'}{s})^i(1 - \frac{s'}{s})^{N_c - i}.$$

Thus, the probability of a particular bivariate polynomial being compromised is $P_{cd} = 1 - \sum_{i=0}^{t} P[i \text{ compromised shares}]$. Since f is any polynomial in $\mathcal{F}$, the fraction of compromised links between non-compromised nodes can be estimated as P_{cd}. Figure 2(a) includes the relationship between the fraction of compromised links for non-compromised nodes and the number of compromised nodes for some combinations of s and s'. We can see that the random subset scheme provides high security guarantee in terms of the fraction of compromised links between non-compromised nodes when the number of compromise nodes does not exceed certain threshold. (To save space, Figure 2 also includes the performance of the basic probabilistic scheme [Eschenauer and Gligor 2002] and the q-composite scheme [Chan et al. 2003], which will be used for the comparison in Section 4.5.)

If an attacker knows the distribution of polynomials over sensor nodes, he/she may target at specific sensor nodes in order to compromise the keys derived from a particular polynomial. In this case, the attacker only needs to compromise $t + 1$ sensor nodes. However, it is generally more difficult than randomly compromising sensor nodes, since the attacker has to compromise the *selected* nodes.

An easy fix to remove the above threat is to restrict that each polynomial be used for at most $t + 1$ times. As a result, an attacker cannot recover a polynomial unless he/she compromises all related sensor nodes. Though effective at improving the security, this method also puts a limit on the maximum number of sensor nodes for a given combination of s and s'. Indeed, given the above constraint, the total number of sensor nodes cannot exceed $\frac{(t+1) \cdot s}{s'}$.

To estimate the probability of any (direct or indirect) key between two non-compromised nodes being compromised, we assume that the network is fully connected and each pair of nodes can establish a direct or indirect key. Thus, among all pairwise keys, there are a fraction p of direct keys and a fraction $1-p$ of indirect keys on average. For each direct key, it has the probability P_{cd} of being compromised. For each indirect key, if the intermediate node and the two polynomials used in the establishment of this key are not compromised, the key is still secure; otherwise, it cannot be trusted. Thus, the probability of an indirect key being compromised can be estimated by $1 - (1 - p_c)(1 - P_{cd})^2$, where $p_c = \frac{N_c}{N}$. Therefore, the probability of any (direct or indirect) key between two non-compromised nodes being compromised can be estimated by

$$P_c = p \times P_{cd} + (1 - p)[1 - (1 - p_c)(1 - P_{cd})^2].$$

Figure 2(b) includes the relationship between the fraction of compromised (direct or indirect) keys for non-compromised nodes and the number of compromised nodes for some combinations of s and s'. We can see that the random subset scheme also provides high security guarantee in terms of the fraction of compromised (direct or indirect) keys between non-compromised nodes when the number of compromise nodes does not exceed a certain threshold.

Two non-compromised sensor nodes may need to re-establish an indirect key when the current pairwise key is compromised. The knowledge of the identities of compromised nodes is generally a difficult problem, which needs deep investigation. However, when such detection mechanism is available and the node compromises are detected, it is always desirable to re-establish the pairwise key. Thus, we assume that the detection of compromised nodes is done through other techniques, and is not considered in this paper.

Assume the source node contacts d neighbor nodes to re-establish an indirect key with the destination node. Among these d nodes, the average number of non-compromised nodes can be estimated by $\frac{d(N-N_c)}{N}$ for simplicity. If one of these non-compromised nodes shares common non-compromised polynomials with both the source and the destination nodes, a new pairwise key can be established. Thus, the probability of re-establishing an indirect key between two non-compromised nodes can be approximately estimated by

$$P_{re} = 1 - [1 - p^2(1 - P_{cd})^2]^{\frac{d(N-N_c)}{N}}.$$

Figure 3 includes the relationship between the probability of re-establishing an indirect key for non-compromised nodes and the number of compromised nodes in the network. It shows that there is still a high probability to re-establish a pairwise key between two non-compromised nodes when the current key is compromised, given that the network still provides certain security performance (e.g. less than 60% compromised links).

4.5 Comparison with Previous Schemes

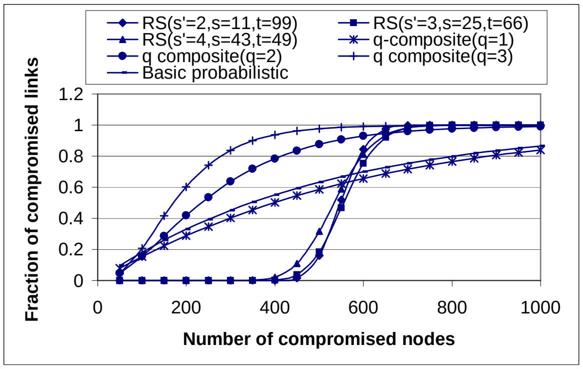
The random subset assignment scheme has a number of advantages compared with the basic probabilistic scheme [Eschenauer and Gligor 2002], the q-composite scheme [Chan et al. 2003], and the random pairwise keys scheme [Chan et al. 2003]. In this analysis, we first compare the communication and computational overheads introduced by these schemes given certain storage constraint, and then compare their security performance under attacks.

We do not compare the random subset assignment scheme with the multiple-space key predistribution scheme in [Du et al. 2003], since these two schemes are actually equivalent to each other. In fact, in multiple-space key predistribution scheme, the elements in the second row of matrix G can be considered as the IDs of sensor nodes in the random subset assignment scheme; each matrix D_i can be considered as the coefficients of a bivariate polynomial; each row in a matrix A_i can be considered as a polynomial share; computing a key through $A_c(i) \times G(j)$ can be considered as evaluating a polynomial share.

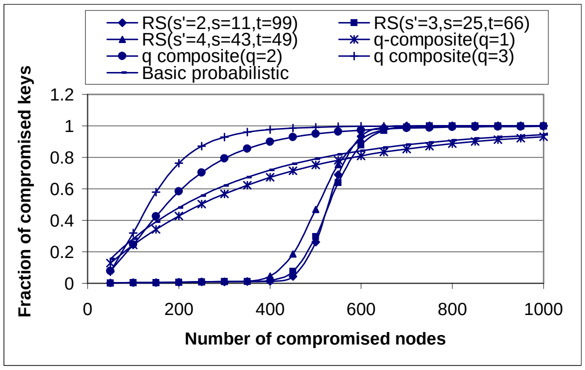
After the direct key establishment, the basic idea of the path key establishment is to find an intermediate node that shares direct keys with both the source and the destination nodes. This is similar in all the previous schemes and the random subset assignment scheme. For simplicity, we focus on the overheads in direct key establishment. Note that each coefficient in our scheme takes about the same amount of space as a cryptographic key, since F_q is a finite field that can just accommodate the keys. We assume that each sensor node can store up to C keys or polynomial coefficients.

The communication and computational overheads for different schemes are summarized in Table I. The communication overhead is calculated using the size of the list of key or polynomial IDs; and the computational overhead is calculated using the number of comparisons in identifying the common key or polynomial, and the number of polynomial evaluations, assuming that the IDs of keys or polynomials are stored in ascend order in each node, and binary search is used to locate the ID of the common key or polynomial.

4.5.1 *Comparison with the Basic Probabilistic and the q-Composite Scheme.* According to Table I, we can see that the random subset assignment is usually much more efficient than the basic probabilistic scheme [Eschenauer and Gligor 2002] and the q-composite



(a) Fraction of compromised links between non-compromised nodes v.s. number of compromised nodes



(b) Fraction of compromised keys (direct or indirect) between non-compromised nodes v.s. number of compromised nodes. Assume $N = 20,000$

Fig. 2. Performance of the random subset assignment scheme under attacks. RS refers to the random subset assignment scheme. Assume each node has available storage for 200 keys and $p = 0.33$.

scheme [Chan et al. 2003] in terms of communication overhead due to small s and s'. Indeed, this overhead is reduced by a factor of at least $t + 1$. However, the computation overhead is more expensive in the random subset assignment scheme, since it has to evaluate a t-degree polynomial.

Figures 2(a) and 2(b) compare the security performance of the random subset assignment scheme with the basic probabilistic scheme [Eschenauer and Gligor 2002] and the q-composite scheme [Chan et al. 2003]. These figures clearly show that before the number of compromised sensor nodes reaches a certain point, the random subset assignment scheme performs much better than both of the other schemes. When the number of compromised

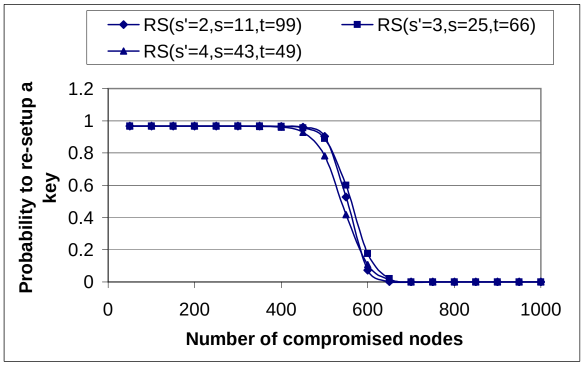


Fig. 3. The probability of re-establishing a pairwise key using path discovery. Assume each node has available storage equivalent to 200 keys, and contacts 30 neighbor nodes $d = 30$. Assume $N = 20,000$

Table I. Communication and computational overheads for direct key establishment in different schemes. s_k is the key pool size in the basic probabilistic scheme and the q-composite scheme. $s' = \frac{C}{t+1}$. The last row will be discussed in Section 5.

	Communication	Computation
Basic probabilistic scheme [Eschenauer and Gligor 2002]	$C \log s_k$	$\frac{2C+p-pC}{2} \log C$ comparisons
q-composite scheme [Chan et al. 2003]	$C \log s_k$	$C \log C$ comparisons
Random pairwise keys scheme [Chan et al. 2003]	0	0
Random subset assignment scheme	$s' \log s$	$\frac{2s'+p-ps'}{2} \log s'$ comparisons + 1 polynomial evaluation
Grid-based scheme	0	1 polynomial evaluation

nodes exceeds a certain point, the other schemes have fewer compromised links or keys than the random subset assignment scheme. Nevertheless, under such circumstances, none of these schemes provide sufficient security due to the large fraction of compromised links (over 60%) or the large fraction of compromised (direct or indirect) keys (over 80%). Thus, the random subset assignment scheme clearly has advantages over the basic probabilistic scheme [Eschenauer and Gligor 2002] and the q-composite scheme [Chan et al. 2003].

4.5.2 *Comparison with the Random Pairwise Keys Scheme.* As shown in Table I, the random pairwise keys scheme [Chan et al. 2003] does not have any communication and computational overheads in direct key establishment, since it stores the IDs of all other nodes with which it can establish direct keys.

In terms of security performance, the random pairwise keys scheme does not allow reuse of the same key by multiple pairs of sensor nodes [Chan et al. 2003]. Thus, the compromise of some sensor nodes does not lead to the compromise of direct keys shared between non-compromised nodes. As we discussed earlier, with a restriction that no polynomial be used more than $t + 1$ times, our scheme can ensure the same property.

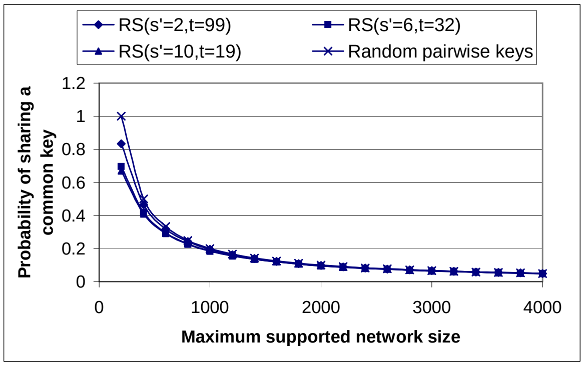


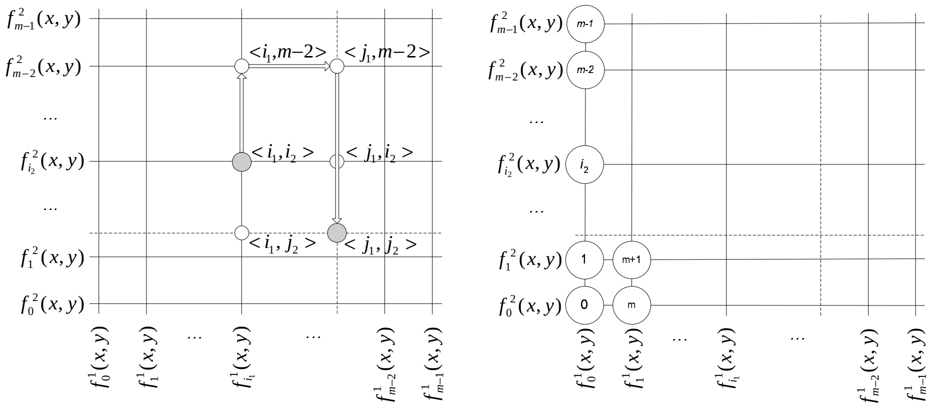
Fig. 4. The relationship between the probability of establishing a common key and the maximum supported network size in order to be resilient against node compromise.

Now we compare the performance between the random subset assignment scheme under the above restriction and the random pairwise keys scheme. The maximum number of nodes that the random subset assignment scheme supports can be estimated as $N = \frac{s \times (t+1)}{s'}$. Assuming the storage overhead in each sensor node is $C = s' \cdot (t+1)$, we have $s = \frac{N \times s'^2}{C}$. Together with Equation 1, we can derive the probability of establishing a direct key between two nodes with a given storage constraint. Figure 4 plots the probability of two sensor nodes sharing a direct key in terms of the maximum network size for the random pairwise keys scheme [Chan et al. 2003] and the random subset assignment scheme under restriction. We can easily see that the random subset assignment scheme under restriction has lower but almost the same performance as the random pairwise keys scheme.

The random subset assignment scheme has several advantages over the random pairwise keys scheme [Chan et al. 2003]. In particular, in the random subset assignment scheme, sensor nodes can be added dynamically without having to contact the previously deployed sensor nodes. In contrast, in the random pairwise keys scheme, if it is necessary to dynamically deploy sensor nodes, the setup server has to either reserve space for sensor nodes that may never be deployed, which reduces the probability that two deployed nodes share a common key, or inform some previously deployed nodes of additional pairwise keys, which introduces additional communication overhead. Moreover, given certain storage constraint, the random subset assignment scheme (without the restriction on the reuse of polynomials) allows the network to grow, while the random pairwise keys scheme has an upper limit on the network size. Thus, the random subset assignment scheme would be a more attractive choice than the random pairwise keys scheme in certain applications.

5. HYPERCUBE-BASED KEY PREDISTRIBUTION

In this section, we give another instantiation of the framework, which we call the *hypercube-based* key predistribution. This scheme has a number of attractive properties. First, it guarantees that any two sensor nodes can establish a pairwise key when there are no compromised sensor nodes, assuming that the nodes can communicate with each other. Second, this scheme is resilient to node compromises. Even if some nodes are compromised, there



(a) An example of hypercube when $n = 2$

(b) An example order of node assignment

Fig. 5. Hypercube-based key predistribution when $n = 2$

is still a high probability to re-establish a pairwise key between two non-compromised nodes. Third, a sensor node can directly determine whether it can establish a direct key with another node, and if it can, which polynomial should be used. As a result, there is no communication overhead during polynomial share discovery.

Note that in the preliminary version of this paper [Liu and Ning 2003b], we studied a key predistribution technique named *grid-based key predistribution*. Hypercube-based key pre-distribution is a generalization of grid-based key predistribution. The grid-based key predistribution scheme is interesting due to its simplicity. However, we do not explicitly discuss it here because of space reasons. Please refer to [Liu and Ning 2003b] for details.

5.1 The Hypercube-Based Scheme

Given a total of N sensor nodes in the network, the hypercube-based scheme constructs an n-dimensional hypercube with m^{n-1} bivariate polynomials arranged for each dimension j, $\{f^j_{\langle i_1,\ldots,i_{n-1}\rangle}(x,y)\}_{0\leq i_1,\ldots,i_{n-1}<m}$, where $m = \lceil \sqrt[n]{N} \rceil$. Figure 5(a) shows a special case of the hypercube-based scheme when $n = 2$ (i.e., the grid-based scheme). In this figure, each column i is associated with a polynomial $f^1_i(x,y)$, and each row i is associated with a polynomial $f^2_i(x,y)$. The setup server then assigns each node in the network to a unique coordinate in this n-dimensional space. For the sensor node at coordinate $(j_1, ..., j_n)$, the setup server predistributes the polynomial shares of $\{f^1_{\langle j_2,\ldots,j_n\rangle}(x,y)$,..., $f^n_{\langle j_1,\ldots,j_{n-1}\rangle}(x,y)\}$ to this node. As a result, sensor nodes can perform share discovery and path discovery using this predistributed information.

For convenience, we encode a node's coordinate in the hypercube into a single-valued node ID. Every valid coordinate in the hypercube is first converted into n l-bit binary strings (one from each dimension), where $l = \lceil \log_2 m \rceil$. These n binary strings are then concatenated together to generate an integer value, which is used as the ID of the node. In our discussion, we conceptually represent each ID j as $\langle j_1, ..., j_n\rangle$, where j_i is called the *sub-index* of ID j in dimension i, which also represents the i^{th} l bits of j.

(1) **Subset assignment:** The setup server randomly generates $n \times m^{n-1}$ t-degree bivariate polynomials $\mathcal{F} = \{f^j_{\langle i_1,...,i_{n-1}\rangle}(x,y) \mid 1 \leq j \leq n, 0 \leq i_1,...,i_{n-1} < m\}$ over a finite field F_q. For each sensor node, the setup server selects an unoccupied coordinate $(j_1,...,j_n)$ in the n-dimensional space and assigns it to this node. This coordinate $\langle j_1,...,j_n\rangle$ is then used as the ID of this node. The setup server then distributes $\{ID, f^1_{\langle j_2,...,j_n\rangle}(j_1,y), ..., f^n_{\langle j_1,...,j_{n-1}\rangle}(j_n,y)\}$ to this sensor node. To facilitate path discovery and guarantee that there is at least one key path exists when there are no compromised nodes and any two nodes can communicate with each other, we always select the coordinate corresponding to the smallest unassigned ID. Specifically, the setup server assigns the i^{th} sensor node that requests for polynomial shares the coordinate $(a_1, a_2, \cdots, a_n)$, where $a_j = \lfloor \frac{i}{m^{n-j}} \rfloor \bmod m^{n-j+1}$ for $1 \leq j \leq n$. Figure 5(b) shows a possible order to assign coordinates to sensor nodes when $n = 2$. It is easy to see that if there exist nodes at $\langle i,j\rangle$ and $\langle i',j'\rangle$, then there must be a node at either $\langle i,j'\rangle$ or $\langle i',j\rangle$, or both.

(2) **Polynomial share discovery:** To establish a pairwise key with node j, node i checks whether they have the same sub-indexes in $n-1$ dimensions. In other words, it checks the Hamming distance d_h between their IDs i and j. If $d_h = 1$, nodes i and j share a common polynomial, and they can establish a direct key using the polynomial-based key predistribution scheme; otherwise, they need to go through path discovery to establish an indirect key. For example, if $j_k = i_k$ for all $1 \leq k \leq n-1$ ($d_h = 1$), both nodes i and j have polynomial shares of $f^n_{\langle j_1,...,j_{n-1}\rangle}(x,y)$, and thus can use this polynomial to establish a direct key.

(3) **Path discovery:** If nodes i and j cannot establish a direct key, they need to find a key path between each other in the hypercube. Indeed, if there are no compromised nodes and any two nodes can communicate with each other, it is guaranteed that there are at least one key path which can be used to establish a session key between any two nodes due to the node assignment algorithm. In fact, nodes i and j can predetermine such a key path using the following key path predetermination algorithm without communicating with others. For example, in Figure 5(a), both of node $\langle i_1, j_2\rangle$ and $\langle j_1, i_2\rangle$ can help node $\langle i_1, i_2\rangle$ establish a pairwise key with node $\langle j_1, j_2\rangle$. Assume $i > j$ if we consider node IDs i and j as integer values. The following algorithm can be performed on either of them.

   (a) The source node maintains a set $\mathcal{L} = \{d_1, ..., d_w\}_{d_1<d_2\cdots<d_w}$ that records the dimensions that nodes i and j have different sub-indexes, a list $\mathcal{P}$ that records the key path computed by this algorithm, a most recently computed intermediate node u and the largest ID having been assigned. Initially, $\mathcal{P}$ is a list with a single node i and $u = i$.

   (b) Given u and $\mathcal{L}$, the next intermediate node v is computed by randomly selecting an element d' in $\mathcal{L}$ so that $v = \langle u_1, \cdots, u_{d'-1}, j_{d'}, u_{d'+1}, \cdots, u_n\rangle$ is not larger than the largest ID having been assigned (or i if this is not available). The algorithm then removes d' from $\mathcal{L}$, appends v to $\mathcal{P}$, and let $u = v$. If $\mathcal{L}$ is empty, it appends j to $\mathcal{P}$ and returns $\mathcal{P}$ as the discovered key path; otherwise it repeats this step.

The correctness of the above key path predetermination algorithm is guaranteed by Lemma 5.1. Once such a key path is computed, the source node i can send a request to the destination node j along the key path to establish an indirect key. For example, if $i = \langle 1,3,5\rangle$ and $j = \langle 0,2,4\rangle$, the key path "$\langle 1,3,5\rangle$, $\langle 0,3,5\rangle$, $\langle 0,2,5\rangle$, $\langle 0,2,4\rangle$" is one of those paths. To establish an indirect key with j, the source node i sends key establishment request to node $\langle 0,3,5\rangle$, node $\langle 0,3,5\rangle$ forwards the request to node $\langle 0,2,5\rangle$, which further forwards the request to the destination node j. Every message transmitted between two adjacent nodes in the key path is encrypted and authenticated with the direct key shared between them.

LEMMA 5.1. *The above key path predetermination algorithm guarantees to compute a key path between any two sensor nodes.*

PROOF. We first show the size of $\mathcal{L}$ is reduced by 1 each time Step (b) is executed. To prove this, we need to prove that given u and $\mathcal{L}$, there exists at least one element d' in $\mathcal{L}$ so that $v = \langle u_1, \cdots, u_{d'-1}, j_{d'}, u_{d'+1}, \cdots, u_n\rangle$ is not larger than the largest ID having been assigned. Note that every $u_{d'}$ is either $i_{d'}$ or $j_{d'}$. Consider d_1. Since $i > j$, we have $i_{d_1} > j_{d_1}$. Thus, if $u_{d_1} = i_{d_1}$, we choose $d' = d_1$ and compute the next node v. It is easy to verify that $v < i$. If $u_{d_1} = j_{d_1}$ (d_1 has been chosen before), we have $u_{d_1} < i_{d_1}$. This implies that $v < i$ for any d' in $\mathcal{L}$. Thus, we can choose any value in $\mathcal{L}$. As a result, u can always find the next node v, and the size of the set $\mathcal{L}$ is reduced by 1 each time Step (b) is executed. Eventually, the above key predetermination algorithm will output a sequence of nodes with node j as the last node. Moreover, the Hamming distance between u and v in the second step is exactly 1. This implies that every two adjacent nodes in $\mathcal{P}$ can establish a direct key. Thus, we can conclude that the above key path predetermination algorithm guarantees to compute a key path between any two sensor nodes. □

5.2 Dynamic Key Path Discovery

Though the path discovery algorithm described above can predetermine a key path with a number of intermediate nodes, the intermediate nodes may have been compromised, or are out of communication range in some situations. However, there are alternative key paths. In particular, we may reuse the predetermined paths at other nodes to find a secure key path. For example, in Figure 5(a), besides node $\langle i_1, j_2\rangle$ and $\langle j_1, i_2\rangle$, node $\langle i_1, m-2\rangle$ has a predetermined path to node $\langle j_1, j_2\rangle$ through node $\langle j_1, m-2\rangle$. Thus, it can help node i setup a common key with node j.

Though it is possible to flood the network to find a key path, the resource constraints on sensor nodes make this method impractical. Instead, we propose the following algorithm to find a key path between nodes S and D dynamically. The basic idea is to have the source node and each intermediate node contact a non-compromised node that is “closer” to the destination node in terms of the Hamming distance between their IDs. Indeed, if there are no compromised nodes in the network, the above key path predetermination algorithm can always find a key path if any two nodes can communicate with each other. In practice, we may use the dynamic path discovery instead to achieve better performance when there are attacks or communication failures. To increase the chance of success, the following algorithm may be performed multiple rounds. It is assumed that every message between two nodes in the algorithm is encrypted and authenticated with the direct key between them.

(1) In order to establish an indirect key with node D, node S generates a random number r and maintains a counter c with initial value 0. In each round, it increments c and computes $K_c = F(r, c)$, where F is a pseudo random function [Goldreich et al. 1986]. Then, it constructs a message $M = \{S, D, K_c, c, flag\}$ with $flag = 1$, and goes to the next step.

(The $flag$ in message M indicates whether the Hamming distance is reduced by forwarding M to the next intermediate node. The purpose is to control the length of the path discovered by this algorithm and the number of messages.)

(2) Consider a sensor node u having the message $M = \{S, D, K_c, c, flag\}$. Node u first tries to find a non-compromised node v that can establish a direct key with u using a non-compromised polynomial and has a smaller Hamming distance to D than u. If this succeeds, u set $flag$ in M to 1 and sends the modified message M to v. We can see that the Hamming distance between v and D is one smaller than that between u and D.

    If u cannot find such a node and $flag$ in M is 0, the path discovery stops. Otherwise, it selects a non-compromised node v that can establish a direct key with u using a non-compromised polynomial and whose Hamming distance to D is the same as u. If u finds such a node v, it sets $flag$ in message M to 0 and sends to v the modified message M. If it cannot find such a node, the path discovery protocol at this node stops.

(3) When the destination node D receives the key establishment request, it knows that node S wants to setup a pairwise key with it. Node D then sets the pairwise key as $K_{S,D} = K_c$, and informs node S the counter value c. As a result, both sensor nodes share the same pairwise key.

LEMMA 5.2. *For any two nodes S and D, the above dynamic key path discovery algorithm guarantees to find a key path with $d_h - 1$ intermediate nodes if there are no compromised nodes and any two nodes can communicate with each other, where d_h is the Hamming distance between S and D.*

PROOF. Let $\mathcal{L} = \{d_1, ..., d_w\}_{d_1<d_2\cdots<d_w}$ records the dimensions in which nodes u and D have different sub-indexes. If $\mathcal{L}$ only has one element, S and D can establish a direct key and the path discovery succeeds.

Now assume $\mathcal{L}$ contains more than one element. We show that any intermediate node u can find a dimension e so that u and D have different sub-indexes in this dimension and the node $\langle u_1, ..., u_{e-1}, D_e, u_{e+1}, ..., u_n\rangle$ exists. Consider dimension d_1. If $u > D$, we have $u_{d_1} > D_{d_1}$. Thus, u can choose node $v = \langle u_1, ..., u_{d_1-1}, D_{d_1}, u_{d_1+1}, ..., u_n\rangle$. Since $v < u$, v must be the ID of an existing node. If $u < D$, we can choose any value in $\mathcal{L}$ other than d_1, since we always have $v < D$ and v must be the ID of an existing node. Thus, if there are no compromised nodes and any two nodes can communicate with each other, any intermediate node will succeed in finding the next closer node v. Since each v has one more common sub-index than the corresponding u, the Hamming distance between each v and D will be smaller than that between u and D by 1. Therefore, there will be $d_h - 1$ intermediate nodes between S and D. □

LEMMA 5.3. *The number of intermediate nodes in the key path discovered in the above dynamic key path discovery algorithm never exceeds $2(d_h - 1)$.*

PROOF. Consider the $flag$ values in the sequence of unicast messages in the path discovery $flag_1, ..., flag_i$. First, it contains at most d_h ones, since otherwise, the request message should have already reached the destination node before the last message containing $flag = 1$. Second, there are no two consecutive zeros in this sequence, since the second step will stop if it cannot find next closer node and the flag in the current message is zero. Third, the last two values ($flag_{i-1}$ and $flag_i$) in this sequence are always 1 for a successful path discovery. Consider the last three nodes in the key path u, v, D, where D is the destination node. It is obvious that $flag_i$ is always 1 for a successful discovery. If $flag_{i-1} = 0$, the Hamming distance between u and D is one, and there is no intermediate node between u and D. Thus, we know that both flag values are 1. Hence, we can conclude that the maximum length of this sequence is $d_h - 2 + d_h - 1 + 2 = 2d_h - 1$. This indicates that the maximum number of intermediate nodes in the key path is $2(d_h - 1)$. □

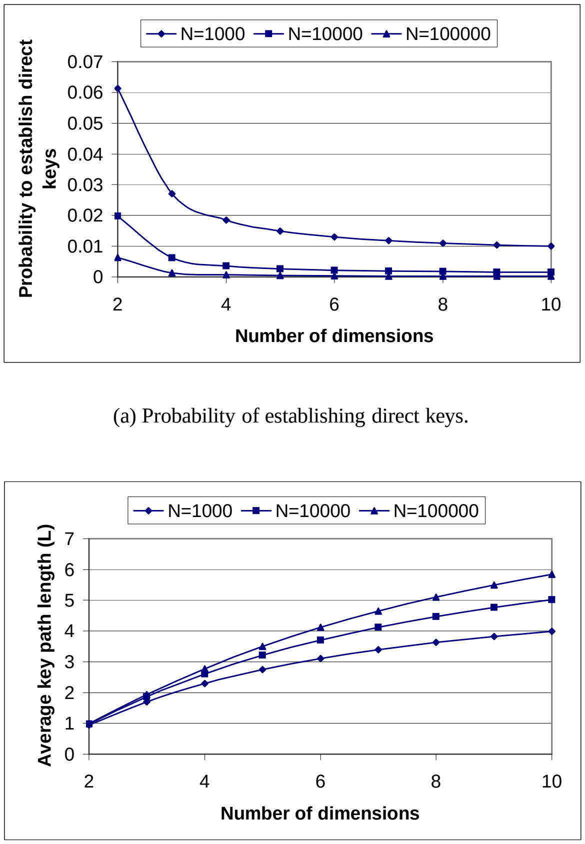
5.3 Performance

Each sensor node stores n polynomial shares and each polynomial is shared by about m different nodes, where $m = \lceil \sqrt[n]{N} \rceil$. Thus, each node can establish direct keys with $n(m-1)$ other nodes. This indicates that the probability to establish direct keys between sensor nodes can be estimated by $\frac{n(m-1)}{m^n-1} = \frac{n(\lceil \sqrt[n]{N} \rceil - 1)}{N-1}$. Figure 6(a) shows that the probability of establishing a direct key between two nodes decreases as the number of dimensions n or the network size N grows. However, according to the path discovery algorithm, if there are no compromised nodes and any two nodes can communicate with each other, it is guaranteed that any two nodes can establish a pairwise key (directly or indirectly).

5.4 Overhead

Each node has to store n polynomial shares and the IDs of revoked nodes with which it can establish direct keys. The former introduces $n(t+1)\log q$ bits storage overhead. For the latter part, a node only needs to remember up to t compromised node IDs for each polynomial, since if $t+1$ shares of one bivariate polynomial are compromised, this polynomial is already compromised and can be discarded. In addition, a sensor node i only needs to remember one sub-index of each revoked ID, because the IDs of node i and the revoked node only differ on one sub-index. Thus, it requires at most ntl bits storage overhead to keep track of revoked node IDs. Totally, the storage overhead at senor nodes is at most $n(t+1)\log q + ntl$ bits, where $l = \lceil \log_2 m \rceil$. To establish a direct key, a sensor node only needs the ID of the other node, and there is no communication overhead. However, if two nodes cannot establish a direct key, they need to find a number of intermediate nodes to help them establish a temporary session key. When key path predetermination is used, discovering a key path does not introduce any communication overhead. However, when dynamic key path discovery is used, this process involves a number of unicast messages. From Lemma 5.3, we know that the dynamic path discovery introduces at most $2d_h - 1$ unicast messages if every unicast message is successfully delivered.

Now let us estimate the communication overhead during the path key establishment, assuming the key path is already discovered. In sensor networks, sending a unicast message between two arbitrary nodes may involve the overhead of establishing a route. However, finding a route in a sensor network depends on routing protocol, which is also required by other schemes to do path discovery. In fact, we are unable to give a precise analysis on this overhead due to the undecided routing protocol. Thus, for simplicity, we use the number of unicast messages to estimate the communication overhead involved in the path key establishment. In fact, there are $L+1$ unicast messages for the path key establishment using a key path with length L if every unicast message is successfully delivered. If there are no compromise sensor nodes and any two nodes can communicate with each other, there exists at least one key path with $d_h - 1$ intermediate nodes, which is indeed one of



(a) Probability of establishing direct keys.

(b) Average key path length

Fig. 6. Performance of the hypercube-based scheme

the the shortest key paths.

Consider two nodes u ($\langle u_1, ..., u_n \rangle$) and v ($\langle v_1, ..., v_n \rangle$). The probability of having $u_e = v_e$ for any $e \in \{1, \cdots, n\}$ is $\frac{1}{m}$, and the probability of having exactly i different sub-indexes is

$$P[i \text{ different sub-indexes}] = \frac{n!}{i!(n-i)!} \frac{1}{m^{n-i}} (1 - \frac{1}{m})^i.$$

Thus, the average key path length can be estimated by

$$L = \sum_{i=1}^{n} (i-1) \times P[i \text{ different sub-indexes}].$$

Figure 6(b) shows the relationship between the the average key path length and the number of dimensions given different network sizes. We can see that the average key path length (and thus the communication overhead) increases as the number of dimensions or the network size grows.

In terms of computational overhead, each unicast message requires one polynomial evaluation, one authentication and one encryption at the source node, and one polynomial evaluation, one authentication and one decryption at the destination node. Since $L+1$ unicast messages are needed for a key path with length L, there are totally $2(L+1)$ polynomial evaluations, $2(L+1)$ authentications, $L+1$ encryptions, and $L+1$ decryptions involved in the path key establishment if every message is successfully delivered.

5.5 Security Analysis

An adversary may launch two types of attacks against the hypercube-based scheme. First, the attacker may target the pairwise key between two particular sensor nodes. The attacker may either try to compromise the pairwise key, or prevent the two sensor nodes from establishing a pairwise key. Second, the attacker may target the entire network to lower the probability or to increase the cost to establish pairwise keys.

5.5.1 *Attacks against a Pair of Nodes.* We focus on the attacks on the communication between two particular sensor nodes u and v without compromising either of them. If they can establish a direct key, the only way to compromise the key without compromising the related nodes is to compromise the shared polynomial between them, which requires the attacker to compromise at least $t+1$ sensor nodes. If they have established an indirect key through path key establishment, the attacker may compromise one of the polynomials or the nodes involved in this establishment, so that the attacker can recover the key if it has the message used to deliver this key. However, even if the attacker compromises the current pairwise key, the related sensor nodes may still re-establish another pairwise key with a different key path. To prevent u from establishing a pairwise key with v, the attacker has to compromise all those n polynomial shares on u (or v) so that node u (or v) is unable to use any polynomial to setup a pairwise key; otherwise, there may still exist non-compromised sensor nodes that can help them establish a new pairwise key. For each of these polynomial shares, the attacker has to compromise at least $t+1$ nodes. This means that the attacker has to compromise at least $n(t+1)$ sensor nodes to prevent u and v from establishing another pairwise key.

5.5.2 *Attacks against the Network.* Since the adversary also knows the distribution of polynomials over sensor nodes, it may systematically attack the network by compromising the polynomials in $\mathcal{F}$ one by one in order to compromise the entire network. Assume the attack compromises b bivariate polynomials. There are up to bm sensor nodes with at least one compromised polynomial share. Among all the remaining $N - bm$ sensor nodes, none of the secure links between them is compromised, since the polynomials used to establish direct keys between them are not compromised. However, the indirect keys in the remaining part of network could be affected, since the common polynomial between two intermediate nodes in the key path may be compromised. Nevertheless, there is still a high probability to re-establish a new indirect key between them, even if an indirect key between two non-compromised nodes is compromised.

Alternatively, the adversary may randomly compromise sensor nodes to attack the path discovery process in order to make it more expensive to establish pairwise keys. In the following, we first investigate the probability of a direct key (secure link) being compromised, and then investigate the probability of any (direct or indirect) key being compromised under node compromises.

Assume a fraction p_c of sensor nodes in the network are compromised. Then, the probability that exactly i shares on a particular bivariate polynomial have been disclosed is

$$P[i \text{ compromised shares}] = \frac{m!}{i!(m-i)!} p_c^i (1-p_c)^{m-i},$$

where $m = \lceil \sqrt[n]{N} \rceil$. Thus, the probability of a particular bivariate polynomial being compromised is $P_{cd} = 1 - \sum_{i=0}^{t} P[i \text{ compromised shares}]$. If $m \gg t+1$, this is equivalent to the probability of any link (direct key) between non-compromised nodes being compromised. For a small m, P_{cd} only represents the fraction of compromised bivariate polynomials. For example, when $n = 4$ and $N = 20,000$, we have $m = 12$ and $t = 11$. In this case, we do not use the fraction of compromised bivariate polynomial to estimate the fraction of compromised links between non-compromised nodes. Instead, we note that the fraction of compromised links between non-compromised nodes in this situation is zero, which implies perfect security against node compromises.

Figure 7(a) shows the relationship between the fraction of compromised links for non-compromised nodes and the fraction of compromised sensor nodes with different number of dimensions.[1] We can see that given the fixed network size and storage overhead, the hypercube-based scheme with more dimensions has higher security performance.

Now let us compute the probability of any (direct or indirect) key between two non-compromised nodes being compromised. Suppose sensor nodes u and v have different sub-indexes in i dimensions. The key path discovered between them involves $i-1$ intermediate nodes and i bivariate polynomials. If none of these $i-1$ intermediate nodes and i bivariate polynomials is compromised, the pairwise key is still secure; otherwise, this key cannot be trusted. This means that the probability of this pairwise key being compromised can be estimated by

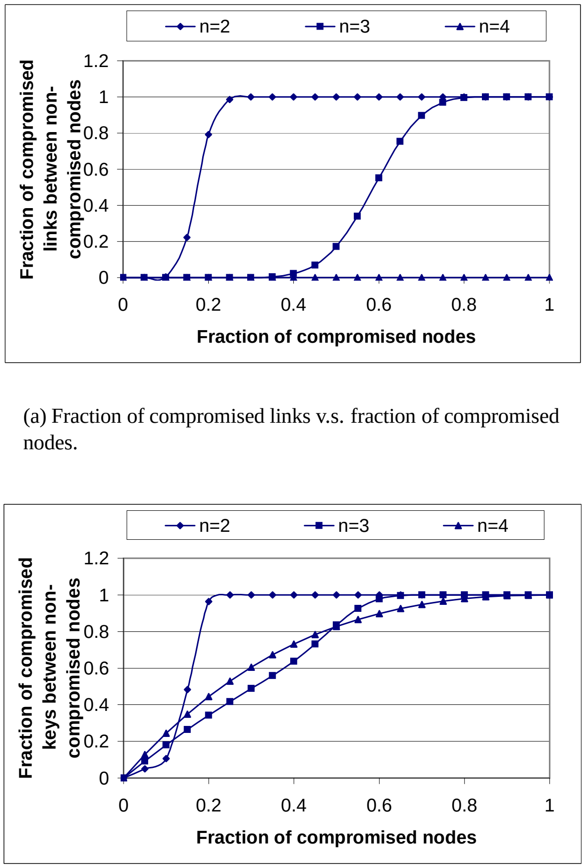
$$P[\text{compromised} \mid i \text{ different sub-indexes}] = 1 - (1-p_c)^{i-1} \times (1-P_{cd})^i.$$

Thus, the probability of any (direct or indirect) key between two non-compromised nodes being compromised can be estimated by

$$P_c = \sum_{i=1}^{n} P[\text{compromised} \mid i \text{ different sub-indexes}] \times P[i \text{ different sub-indexes}].$$

Figure 7(b) shows the relationship between the probability P_c and the fraction of compromised nodes for different number of dimensions. We can still see the improvements of the security when we have more dimensions. This is because the probability of a polynomial being compromised decreases quickly when we have more dimensions. We also note that when the fraction of compromised sensor nodes is less than a certain threshold, having more dimensions decreases the security of the scheme. The reason is that having more dimensions increases the average key path length, which in turn increases the probability of at least one intermediate node in the key path being compromised.

[1] We assume there are totally $20,000$ sensor nodes in the following simulations. Thus, for a 4-dimensional hypercube, we have $m = \lceil \sqrt[4]{20,000} \rceil = 12$. This means that the degree of bivariate polynomial t is not necessary to be larger than 11, and the sensor node needs to store at most $4 \times (11+1) = 48$ coefficients. Therefore, we assume the storage constraint at sensor nodes is equivalent to store 50 keys instead of 200 keys in the analysis of the earlier schemes.



(a) Fraction of compromised links v.s. fraction of compromised nodes.

(b) Fraction of compromised direct and indirect keys v.s. fraction of compromised nodes.

Fig. 7. Security performance of the hypercube-based scheme. Assume each sensor has available storage equivalent to 50 keys, $N = 20,000$, $m = \sqrt[n]{N}$, and $t = \lfloor \frac{50}{n} - 1 \rfloor$.

5.5.3 *Maximum Supported Network Size.* Let us consider the maximum supported network size when the perfect security against node compromises is required. Figure 8 shows the maximum supported network size as a function of the number of dimensions given a fixed memory constraint and guarantee of perfect security against node compromises. We can see that the maximum supported network size increases dramatically when we have more dimensions within the range shown in the figure. (Note that once the number of dimension passes a certain threshold, this maximum supported network size will start to drop.) Indeed, when the number of dimensions is smaller, the hypercube-based scheme can support a larger network by adding more dimensions without increasing the storage overhead or sacrificing the security performance.

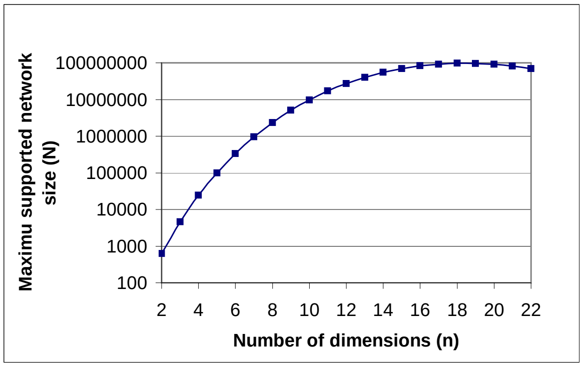


Fig. 8. Maximum supported network size for different number of dimensions. Assume each sensor has available storage equivalent to 50 keys.

5.5.4 *Probability of Re-Establishing a Pairwise Key.* The following analysis estimates the probability to re-establish an indirect key between two non-compromised nodes with the dynamic path discovery algorithm when all predetermined key paths cannot be used due to compromised intermediate nodes or communication failures.

Let I_i denotes the probability to establish a pairwise key between two non-compromised nodes having different sub-indexes in i different dimensions (i.e., the Hamming distance between the two node IDs is $d_h = i$). For a particular node u, we refer to a non-compromised intermediate node as its *closer* node to the destination node if this node can establish a direct key with node u using a non-compromised polynomial and is closer to destination node in terms of the Hamming distance between their IDs. According to the dynamic key path discovery algorithm, the pairwise key can be established if either of the following two cases is true. In the first case, the source node finds a closer node and the selected closer node finds a key path to the destination node. This probability can be estimated by

$$P_1 = \left[1 - [1 - (1 - P_{cd})(1 - p_c)]^i\right] I_{i-1}.$$

In the second case, the source node cannot find any closer node but can establish a direct key using a non-compromised polynomial with a non-compromised node that is able to find a closer node that can find a key path to the destination node. This probability can be estimated by

$$P_2 = (1 - P_1)(1 - P_{cd}^i)\left[1 - [1 - (1 - P_{cd})(1 - p_c)]^{i-1}\right] I_{i-1}.$$

Overall, we have $I_i = P_1 + P_2$ for $i > 1$ and $I_1 = 1 - P_{cd}$. Therefore, the probability of re-establishing an indirect key between two non-compromised nodes can be estimated by

$$P_{re} = \sum_{i=1}^{n} I_i \times P[i \text{ different sub-indexes}].$$

Figure 9 shows this probability for different fractions of compromised sensor nodes. It shows that even if a pairwise key is compromised, there is still a high probability to re-establish a new key if the compromised nodes are detected. In addition, we note that

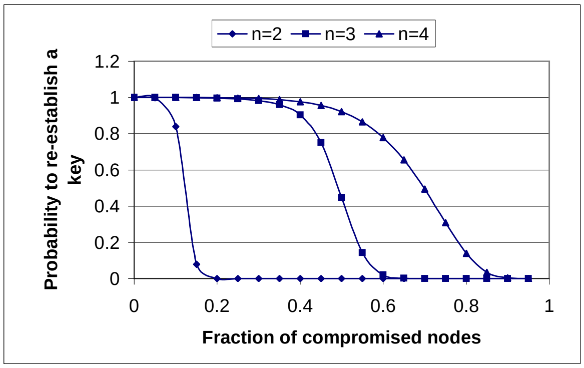


Fig. 9. Probability to re-establish a pairwise key between non-compromised nodes v.s. the fraction of compromised nodes. Assume each sensor node has available storage equivalent to 50 keys, and $N = 20,000$.

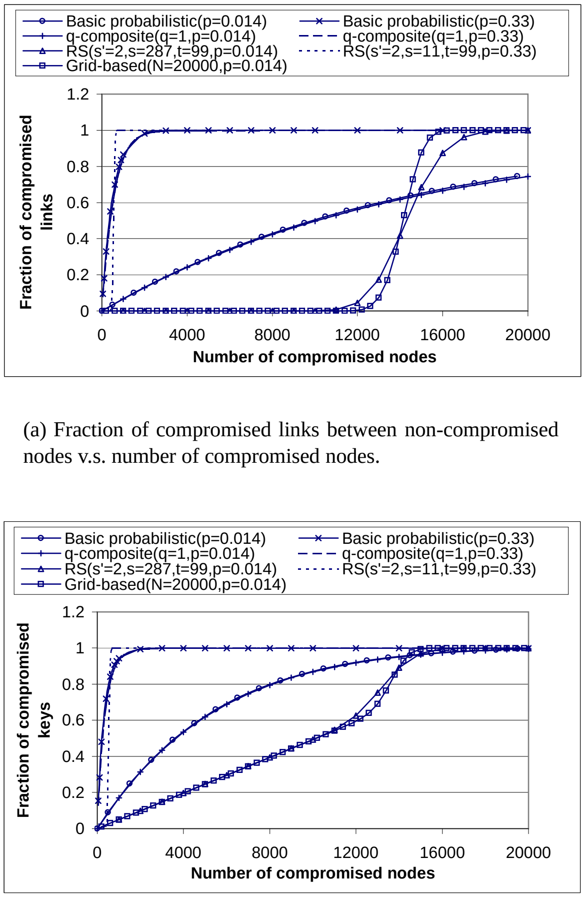
the probability of re-establishing a key increases when we have more dimensions. This is because the probability of a polynomial being compromised decreases quickly as the number of dimensions grows.

5.6 Comparison with Previous Schemes

This subsection compares the hypercube-based key predistribution scheme with the basic probabilistic scheme [Eschenauer and Gligor 2002], the q-composite scheme [Chan et al. 2003], the random pairwise keys scheme [Chan et al. 2003], and the random subset assignment scheme presented in Section 4. In the comparison, we use the hypercube-based scheme with $n = 2$ (i.e., the grid-based scheme) for simplicity.

The communication and computational overheads for direct key establishment in the grid-based scheme and the other schemes are summarized in Table I. We can see that the grid-based scheme is generally much more efficient than the basic probabilistic scheme [Eschenauer and Gligor 2002], the q-composite scheme [Chan et al. 2003], and the random subset assignment scheme in terms of the communication and computational overhead. Compared with the random pairwise keys scheme [Chan et al. 2003], it only involves one more polynomial evaluation, which can be done very efficiently by using the optimization technique in Section 6.

To compare the security between different schemes, we assume the network size is $N = 20,000$, and each node can store up to 200 keys or polynomial coefficients. In the grid-based scheme, we have $m = 142$ and $p = 0.014$. The four curves in the right part of Figure 10(a) and 10(b) show the fraction of compromised links and the fraction of compromised (direct or indirect) keys between non-compromised nodes as a function of the number of compromised sensor nodes given $p = 0.014$. Similar to the comparison in Section 4, the random subset assignment scheme and the grid-based scheme perform much better when there are a small number of compromise nodes. In fact, these two scheme always have better performance when the number of compromised nodes is less than 14,000. When there are more than 14,000 compromised nodes, none of the schemes can provide sufficient security because of the large fraction of compromised links (over 60% compromised links) or the large fraction of compromised (direct or indirect) keys (over 90% compromised



(a) Fraction of compromised links between non-compromised nodes v.s. number of compromised nodes.

(b) Fraction of compromised (direct or indirect) keys between non-compromised nodes v.s. number of compromised nodes.

Fig. 10. Performance of the grid-based key predistribution scheme under attacks. Assume each sensor node has available storage equivalent to 200 keys.

keys).

Though $p = 0.014$ is acceptable for the grid-based scheme, for the basic probabilistic, the q-composite, and the random subset assignment schemes, p should be large enough to make sure the whole network is fully connected. Assume $p = 0.33$. This requires about 42 neighbor nodes for each sensor node to make sure the whole network with 20,000 nodes is connected with a high probability. The three curves in the left part of Figure 10(a) and 10(b) show the fraction of compromised links and the fraction of compromised (direct or indirect) keys between non-compromised nodes as a function of the number of compromised sensors for the above three schemes when $p = 0.33$. We can see that a small number of compromised nodes reveal a large fraction of secrets in the network in these schemes; however, the fraction of compromised links and the fraction of compromised (direct or indirect) keys are much lower in the grid-based scheme for the same number of compromised nodes.

To compare with the random pairwise keys scheme [Chan et al. 2003], we set $m = t+1$, so that the grid-based scheme can provide the same degree of perfect security guarantee as the random pairwise keys scheme. Assume the storage overhead on sensor nodes is $2(t+1) = 2m$. The grid-based scheme can support a network with m^2 nodes, and the probability that two sensor nodes share a direct key is $p = \frac{2}{m+1}$. With the same number sensor nodes and storage overhead, the random pairwise keys scheme [Chan et al. 2003] has $p = \frac{2m}{m^2} = \frac{2}{m}$, which is approximately the same as our scheme.

In addition to the above comparisons, the grid-based scheme has some unique properties that the other schemes do not provide. First, when there is no compromised sensor nodes in the network, it is guaranteed that any pair of sensor nodes can establish a pairwise key either directly without communication, or through the help of an intermediate node when the sensor nodes can communicate with each other. Besides the efficiency in determining the key path, the communication overhead is substantially lower than the previous schemes, which requires real-time path discovery even in normal situations. Second, even if there are compromised sensor nodes in the network, there is still a high probability that two non-compromised sensor nodes can re-establish a pairwise key. Our earlier analysis indicates that it is very difficult for the adversary to prevent two non-compromised nodes from establishing a pairwise key. Finally, due to the orderly node assignment, this scheme allows optimized deployment of sensor nodes so that the sensor nodes that can establish direct keys are close to each other, thus greatly decreasing the communication overhead in path key establishment.

6. IMPLEMENTATION AND EVALUATION

We have implemented the random subset assignment scheme and the grid-based scheme [2] on MICA2 motes [Crossbow Technology Inc.] running TinyOS [Hill et al. 2000], which is an operating system for networked sensor nodes. These implementations were written in nesC [Gay et al. 2003], a C-like programming language used to develop TinyOS and its applications. A critical component in our implementations is the algorithm to evaluate a t-degree polynomial, which is used to compute pairwise keys. In the following, we first present an optimization technique for polynomial evaluation on sensor nodes, and then report the evaluation of this optimization technique and our key predistribution schemes.

6.1 Optimization of Polynomial Evaluation on Sensor Nodes

Evaluating a t-degree polynomial is essential in the computation of a pairwise key in our schemes. This requires t modular multiplications and t modular additions in a finite filed F_q, where q is a prime number that is large enough to accommodate a cryptographic key. This implies that q should be at least 64 bit long for typical cryptosystems such as RC5. However, processors in sensor nodes usually have much smaller word size. For example, ATmega128, which is used in many types of sensors, only supports 8-bit multiplications and has no division instruction. Thus, in order to use the basic scheme, sensor nodes have to implement some large integer operations.

[2] These implementations are included in our *tiny key management* (TinyKeyMan) package, which is available online at `http://discovery.csc.ncsu.edu/software/TinyKeyMan`.

Nevertheless, in our schemes, polynomials can be evaluated in much cheaper ways than polynomial evaluation in general. This is mainly due to the observation that the points at which the polynomials are evaluated are sensor IDs, and these IDs can be chosen from a different finite field $F_{q'}$, where q' is a prime number that is larger than the maximum number of sensors but much smaller than a typical q.

During the evaluation of a polynomial $f(x) = a_t x^t + a_{t-1}x^{t-1} + \cdots + a_0$, since the variable x is the ID of a sensor, the modular multiplication is always performed between an integer in F_q and another integer in $F_{q'}$. For example, to compute the product of two 64-bit integers on a 8-bit CPU, it takes 64 word multiplications with the standard large integer multiplication algorithm, and 27 word multiplications with the Karatsuba-Ofman algorithm [Knuth 1997]. In contrast, it only takes 16 word multiplications with the standard algorithm to compute the product of a 64-bit integer and a 16-bit integer on the same platform. Similarly, reduction of the later product (which is an 80-bit integer) modulo a 64-bit prime is also about 75% cheaper than the former product (which is a 128-bit integer).

Considering the lack of division instruction in typical sensor processors, we further propose to use q' in the form of $q' = 2^k + 1$. Because of the special form of $q' = 2^{16} + 1$, no division operation is needed to compute modular multiplications in $F_{q'}$ [Stallings 1999]. Two natural choices of such prime numbers are $257 = 2^8 + 1$ and $65,537 = 2^{16} + 1$. Using the random subset assignment scheme, they can accommodate up to 256 and 65,536 sensors, respectively. Using the grid-based scheme, they can accommodate up to $256^2 = 65,536$ and $65,536^2 = 4,294,967,296$ sensors, respectively.

To take full advantage of the special form of q', we propose to adapt the basic polynomial-based key predistribution in Section 2 so that a large key is split into pieces and each piece is distributed to sensors with a polynomial over $F_{q'}$. The same technique can be easily applied to all polynomial pool-based schemes with slight modification.

Assume each cryptographic key is n bits. The setup server divides the n-bit key into r pieces of l-bit segments, where $l = \lfloor \log_2 q' \rfloor$ and $r = \lceil \frac{n}{l} \rceil$. For simplicity, we assume $n = l \cdot r$. The setup server randomly generates r t-degree bivariate polynomials $\{f_v(x,y)\}_{v=1,\cdots,r}$ over $F_{q'}$ such that $f_v(x,y) = f_v(y,x)$ for $v = 1, \cdots, r$. The setup server then gives the corresponding polynomial shares on these r polynomials to each sensor node. Specifically, each sensor node i receives $\{f_v(i,x)\}_{v=1,\cdots,r}$. With the basic scheme, each of these r polynomials can be used to establish a common secret between a pair of sensors. These sensors then choose the l least significant bits of each secret value as a key segment. The final pairwise key can simply be the concatenation of these r key segments.

It is easy to verify that this method requires the same number of word multiplications as the earlier one; however, because of the special form of q', no division operation is necessary in evaluating the polynomials. This can significantly reduce the computation on processors that do not have any division instruction.

The security of this scheme is guaranteed by Lemma 6.1.

LEMMA 6.1. *In the adapted key predistribution scheme, the entropy of the key for a coalition of no more than t other sensor nodes is* $r \cdot [\log_2 q' - (2 - \frac{2^{l+1}}{q'})]$, *where* $l = \lfloor \log_2 q' \rfloor$ *and* $r = \lceil \frac{n}{l} \rceil$.

PROOF. Assume that nodes u and v need to establish a pairwise key. Consider a coalition of no more than t other sensor nodes that tries to determine this pairwise key. According to the security proof of the basic key predistribution scheme [Blundo et al. 1993], the entropy of the shared secret derived with any polynomial is $\log q'$ for the coalition. That is, any value from the finite field $F_{q'}$ is a possible value of each of $\{f_j(u,v)\}_{j=1,\ldots,r}$ for the coalition. Since each piece of key consists of the last $l = \lfloor \log_2 q' \rfloor$ bits of one of the above values, values from 0 to $q' - 2^l - 1$ have the probability $\frac{2}{q'}$ to be chosen, while the values from $q' - 2^l$ to $2^l - 1$ have the probability $\frac{1}{q'}$ to be chosen. Denote all the information that the coalition knows as C. Thus, for the coalition, the entropy of each piece of key segment K_j, $j = 1, \cdots, r$, is

$$\begin{aligned} H(K_j|C) &= \sum_{i=0}^{q'-2^l-1} \frac{2}{q'} \log_2 \frac{q'}{2} + \sum_{i=q'-2^l}^{2^l-1} \frac{1}{q'} \log_2 q' \\ &= \frac{2(q'-2^l)}{q'} \log_2 \frac{q'}{2} + \frac{2^{l+1}-q'}{q'} \log_2 q' \\ &= \log_2 q' - (2 - \frac{2^{l+1}}{q'}) \end{aligned}$$

Because the r pieces of key segments are distributed individually and independently, the entropy of the pairwise key for the coalition is

$$H(K|C) = \sum_{j=1}^{r} H(K_j|\cdot) = r \cdot [\log_2 q' - (2 - \frac{2^{l+1}}{q'})].$$

□

Consider a 64-bit key. If we choose $q' = 2^{16} + 1$, the entropy of a pairwise key for a coalition of no more than t compromised sensor nodes is $4 \times [\log_2(2^{16}+1) - (2 - \frac{2^{17}}{2^{16}+1})] = 63.9997$ bits. If we choose $q' = 2^8+1$, this entropy is then $8 \times [\log_2(2^8+1) - (2 - \frac{2^9}{2^8+1})] = 63.983$ bits. Thus, the adapted scheme still provides sufficient security despite of the minor leak of information.

6.2 Evaluation

We first evaluate the performance of our optimization technique for polynomial evaluation. This optimization forms the basis of pairwise key computation in our implementation. We provide two options for this component: one with $q' = 2^8 + 1$ and the other with $q' = 2^{16} + 1$. The typical length of a cryptographic key in resource constrained sensor nodes is 64-bit long. To compute a 64-bit pairwise key, a sensor node has to evaluate 8 t-degree polynomials if $q' = 2^8 + 1$ and 4 t-degree polynomials if $q' = 2^{16} + 1$. The code sizes for the implementations of these two options are shown in Table II. The bytes needed for polynomial coefficients are not included in the code size calculation, since it depends on the applications. Obviously, these two implementations occupy only a small amount of memory at sensor nodes.

The cost of our optimization technique in computing a 64-bit cryptographic key on a MICA2 mote [Crossbow Technology Inc.] is shown in Figure 11, which also includes the cost of generating a 64-bit MAC (Message Authentication Code) for a 64-bit long message using RC5 [Rivest 1994] and SkipJack [NIST 1998] with a 64-bit long key. These two symmetric cryptographic techniques are generally believed to be practical and efficient for

Table II. Code sizes for our optimized polynomial evaluation schemes.

Scheme	ROM (bytes)	RAM (bytes)
$q' = 2^8 + 1$	288	11
$q' = 2^{16} + 1$	416	20

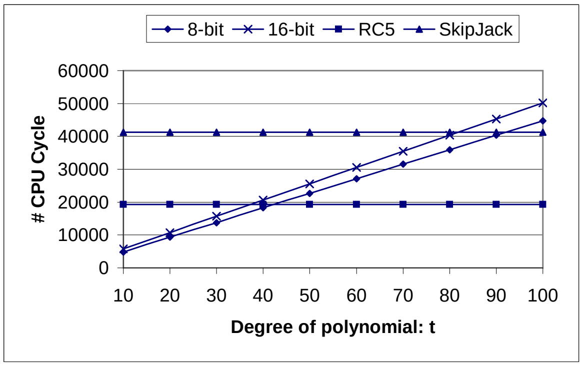


Fig. 11. Comparison with RC5 and SkipJack.

sensor networks. The result shows that computing a pairwise key in our schemes can be faster than generating a MAC using RC5 or SkipJack for a reasonable polynomial degree t; while the value of t is not necessary to be a very large number in practice due to the storage and security concerns, which can be seen from previous analysis in Section 4 and Section 5. The result demonstrates the practicality and efficiency of our proposed schemes.

Table III. The code size for random subset assignment and grid-based scheme. The storage for the polynomial coefficients and the list of compromised nodes are not included in the calculation of code size.

Scheme	ROM (bytes)	RAM (bytes)
Random Subset Assignment	2824	106
Grid-Based	1160	67

According to the result in Figure 11, the 16-bit version is slightly slower than the 8-bit version. However, the 16-bit version can accommodate a lot more sensor nodes than the 8-bit version. Thus, we use the 16-bit option for both the random subset assignment scheme and the grid-based scheme. The code sizes for these two schemes are shown in Table III, which only includes the size of code loaded on sensor nodes, since the components for the setup server are not installed on sensor nodes. In fact, the setup server is not necessary to be a sensor node. We can see that the code size for the grid-based scheme is much smaller than that for the random subset assignment scheme, since the grid-based scheme can directly determine the direct key shared or the key path involved; while the random subset assignment scheme has to contact other nodes and maintain a lot more states.

Together with the analysis in previous sections and the evaluation of computational and storage cost, we can conclude that our schemes are practical and efficient for the current generation of sensor networks.

7. RELATED WORK

Our schemes are based on the polynomial-based key predistribution protocol in [Blundo et al. 1993]. The protocol in [Blundo et al. 1993] was intended to distribute group keys, and is generally not feasible in sensor networks. Our schemes only use the two-party case of this protocol; by enhancing the basic polynomial-based scheme with other techniques such as polynomial pool, our schemes can achieve performance beyond the basic protocol.

Eschenauer and Gligor [Eschenauer and Gligor 2002] proposed a probabilistic key predistribution technique to bootstrap the initial trust between sensor nodes. The main idea is to have each sensor randomly pick a set of keys from a key pool before deployment. Then, in order to establish a pairwise key, two sensor nodes only need to identify the common keys that they share. Chan et al. further extended this idea and propose the q-composite key predistribution [Chan et al. 2003]. This approach allows two sensor nodes to setup a pairwise key only when they share at least q common keys. Chan et al. also developed a random pairwise keys scheme to defeat node capture attacks. In our analysis in earlier Sections, we have demonstrated that our techniques have significant advantages over these schemes. Pietro et al. proposed a seed-based key deployment strategy to simplify the key discovery procedure, and a cooperative protocol to enhance its performance [Pietro et al. 2003].

Du et al. independently discovered a technique equivalent to one of our proposed schemes (random subset assignment) in this paper [Du et al. 2003]. It was published at the same time in the same conference as the preliminary version of this paper [Liu and Ning 2003b]. However, our paper proposes a more general framework, which makes it possible to discover novel key predistribution schemes. Based on this framework, we developed two efficient instantiations. In addition, we also provide the implementation of proposed schemes and include the real experiment results, which demonstrate the practicality and efficiency of our techniques in real applications. The idea of using sensor's pre-deployment knowledge was also independently proposed in [Du et al. 2004] and [Liu and Ning 2003c] to improve the performance of pairwise key predistribution.

There are many other related works in sensor network security. Stajano and Anderson discussed bootstrapping trust between devices through location limited channels such as physical contact [Stajano and Anderson 1999]. Carman, Kruus, and Matt studied the performance of a number of key management approaches in sensor network on different hardware platform [Carman et al. 2000]. Wong and Chan proposed to reduce the computational overhead for key exchange in low power computing device with the help of a more power server [Wong and Chan 2001].

Perrig et al. developed a security architecture for sensor networks, which includes SNEP, a security primitive building block, and μTESLA [Perrig et al. 2001], an adaption of TESLA [Perrig et al. 2000; 2001]. In our previous work, we proposed a multi-level key chain method for the initial commitment distribution in μTESLA [Liu and Ning 2003a]. Basagni et al. presented a key management scheme to secure the communication by periodically updating the symmetric keys shared by all sensor nodes [Basagni et al. 2001]. However, this scheme assumes a tamper-resistant device to protect the key, which is not always available in sensor networks. Zhu et al. proposed a protocol suite named LEAP (Localized Encryption and Authentication Protocol) to help establish individual keys between sensors and a base station, pairwise keys between sensors, cluster keys within a local area, and a group key shared by all nodes [Zhu et al. 2003].

Deng, Han and Mishra developed a collection of mechanisms to improve the security for in-networking processing [Deng et al. 2003]. Zhu et al. proposed an interleaved hop-by-hop authentication to defeat malicious data injection in sensor networks [Zhu et al. 2004]. The problem of secure data aggregation in the presence of compromised nodes are studied in [Hu and Evans 2003a] and [Przydatek et al. 2003]. Our key predistribution schemes can be used to address the key management issues in these techniques.

Wood and Stankovic identified a number of DOS attacks in sensor networks [Wood and Stankovic 2002]. Karlof and Wagner pointed out security goals for routing in sensor networks and analyzed the vulnerabilities as well as the countermeasures for a number of existing routing protocols [Karlof and Wagner 2003]. Sastry, Shankar and Wagner proposed a location verification technique based on the round trip time (RTT) [Sastry et al. 2003] to detect false location claims. Hu and Evans proposed to use directional antenna to detect wormhole attacks in wireless Ad Hoc networks [Hu and Evans 2003b]. Newsome et al. studied the Sybil attack in sensor networks where a node illegitimately claims multiple identities, and developed techniques to defend against this attack [Newsome et al. 2004]. Our proposed techniques can be combined with these techniques to further enhance the security in sensor networks.

8. CONCLUSION AND FUTURE WORK

In this paper, we developed a general framework for polynomial pool-based pairwise key predistribution in sensor networks based on the basic polynomial-based key predistribution in [Blundo et al. 1993]. This framework allows study of multiple instantiations of possible pairwise key establishment schemes. Based on this framework, we developed two specific key predistribution schemes: the random subset assignment scheme and the hypercube-based key predistribution scheme. Our analysis of these schemes indicate that both schemes have significant advantages over the existing approaches. The implementation and experimental results also demonstrate the practicality and efficiency in the current generation of sensor networks.

Several research directions are worth investigating. First, we observe sensor node have low mobility in many applications. Thus, it may be desirable to develop location-sensitive key predistribution techniques to improve the probability for neighbor nodes to share common keys and at the same reduce the threat of compromised nodes. Second, it is critical to detect and/or revoke compromised nodes from an operational sensor network. Third, the hypercube-based key predistribution scheme has some nice properties in the artificial hypercube with which the sensor nodes are organized. We will study ways to map such structures to sensors' physical locations using reasonable deployment models.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments.

REFERENCES

BASAGNI, S., HERRIN, K., BRUSCHI, D., AND ROSTI, E. 2001. Secure pebblenets. In *Proceedings of ACM International Symposium on Mobile ad hoc networking and computing*. 156–163.

BLUNDO, C., DE SANTIS, A., HERZBERG, A., KUTTEN, S., VACCARO, U., AND YUNG, M. 1993. Perfectly-secure key distribution for dynamic conferences. In *Advances in Cryptology – CRYPTO '92, LNCS 740*. 471–486.

CARMAN, D., KRUUS, P., AND B.J.MATT. 2000. Constrains and approaches for distributed sensor network security. Tech. rep., NAI Labs.

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

CROSSBOW TECHNOLOGY INC. Wireless sensor networks. `http://www.xbow.com/Products/Wireless_Sensor_Networks.htm`. Accessed in February 2004.

DENG, J., HAN, R., AND MISHRA, S. 2003. Security support for in-network processing in wireless sensor networks. In *2003 ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN '03)*.

DU, W., DENG, J., HAN, Y. S., CHEN, S., AND VARSHNEY, P. 2004. A key management scheme for wireless sensor networks using deployment knowledge. In *Proceedings of IEEE INFOCOM'04*.

DU, W., DENG, J., HAN, Y. S., AND VARSHNEY, P. 2003. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*. 42–51.

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.

GAY, D., LEVIS, P., VON BEHREN, R., WELSH, M., BREWER, E., AND CULLER, D. 2003. The nesC language: A holistic approach to networked embedded systems. In *Proceedings of Programming Language Design and Implementation (PLDI 2003)*.

GOLDREICH, O., GOLDWASSER, S., AND MICALI, S. 1986. How to construct random functions. *Journal of the ACM 33,* 4 (October), 792–807.

HILL, J., SZEWCZYK, R., WOO, A., HOLLAR, S., CULLER, D., AND PISTER, K. S. J. 2000. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*. 93–104.

HU, L. AND EVANS, D. 2003a. Secure aggregation for wireless networks. In *Workshop on Security and Assurance in Ad Hoc Networks*.

HU, L. AND EVANS, D. 2003b. Using directional antennas to prevent wormhole attacks. In *Proceedings of the 11th Network and Distributed System Security Symposium*. 131–141.

KARLOF, C. AND WAGNER, D. 2003. Secure routing in wireless sensor networks: Attacks and countermeasures. In *Proceedings of 1st IEEE International Workshop on Sensor Network Protocols and Applications*.

KNUTH, D. 1997. *The Art of Computer Programming,* Third Edition ed. Vol. Vol. 2: Seminumerical Algorithms. Addison-Wesley. ISBN: 0-201-89684-2.

LIU, D. AND NING, P. 2003a. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In *Proceedings of the 10th Annual Network and Distributed System Security Symposium*. 263–276.

LIU, D. AND NING, P. 2003b. Establishing pairwise keys in distributed sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*. 52–61.

LIU, D. AND NING, P. 2003c. Location-based pairwise key establishments for static sensor networks. In *2003 ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN '03)*. 720082.

NEWSOME, J., SHI, R., SONG, D., AND PERRIG, A. 2004. The sybil attack in sensor networks: Analysis and defenses. In *Proceedings of IEEE International Conference on Information Processing in Sensor Networks (IPSN 2004)*.

NIST. 1998. Skipjack and KEA algorithm specifications. `http://csrc.nist.gov/encryption/skipjack/skipjack.pdf`.

PERRIG, A., CANETTI, R., SONG, D., AND TYGAR, D. 2000. Efficient authentication and signing of multicast streams over lossy channels. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*.

PERRIG, A., CANETTI, R., SONG, D., AND TYGAR, D. 2001. Efficient and secure source authentication for multicast. In *Proceedings of Network and Distributed System Security Symposium*.

PERRIG, A., SZEWCZYK, R., WEN, V., CULLER, D., AND TYGAR, D. 2001. SPINS: Security protocols for sensor networks. In *Proceedings of Seventh Annual International Conference on Mobile Computing and Networks*.

PIETRO, R. D., MANCINI, L. V., AND MEI, A. 2003. Random key assignment for secure wireless sensor networks. In *2003 ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN '03)*.

PRZYDATEK, B., SONG, D., AND PERRIG, A. 2003. SIA: Secure information aggregation in sensor networks. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys '03)*.

RIVEST, R. 1994. The RC5 encryption algorithm. In *Proceedings of the 1st International Workshop on Fast Software Encryption*. Vol. 809. 86–96.

SASTRY, N., SHANKAR, U., AND WAGNER, D. 2003. Secure verification of location claims. In *ACM Workshop on Wireless Security*.

STAJANO, F. AND ANDERSON, R. 1999. The resurrecting duckling: security issues for ad hoc networks. In *Proceedings of the 7th International Workshop on Security Protocols*. 172–194.

STALLINGS, W. 1999. *Cryptography and Network Security: Principles and Practice*, 2nd ed. Prentice Hall.

WONG, D. AND CHAN, A. 2001. Efficient and mutually authenticated key exchange for low power computing devices. In *Proceedings of ASIA CRYPT 2001*.

WOOD, A. D. AND STANKOVIC, J. A. 2002. Denial of service in sensor networks. *IEEE Computer 35*, 10, 54–62.

ZHU, S., SETIA, S., AND JAJODIA, S. 2003. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*. 62–72.

ZHU, S., SETIA, S., JAJODIA, S., AND NING, P. 2004. An interleaved hop-by-hop authentication scheme for filtering false data in sensor networks. In *Proceedings of 2004 IEEE Symposium on Security and Privacy*.