Establishing Pairwise Keys in Distributed Sensor Networks

Donggang Liu
Cyber Defense Laboratory
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8207
dliu@unity.ncsu.edu

Peng Ning
Cyber Defense Laboratory
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8207
ning@csc.ncsu.edu

ABSTRACT

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 sensors, it is infeasible to use traditional key management techniques such as public key cryptography and key distribution center (KDC). To facilitate the study of novel pairwise key predistribution techniques, this paper presents a general framework for establishing pairwise keys between sensors on the basis of a polynomial-based key predistribution protocol [2]. This paper then presents two efficient instantiations of the general framework: a *random subset assignment* key predistribution scheme and a *grid-based* key predistribution scheme. The analysis in this paper indicates that these two schemes have a number of nice properties, including high probability (or guarantee) to establish pairwise keys, tolerance of node captures, and low communication overhead. Finally, this paper presents a technique to reduce the computation at sensors required by these schemes.

Categories and Subject Descriptors

C.2.0 [**Computer-communication networks**]: General–security and protection

General Terms

Design, Security

Keywords

key management, sensor networks, probabilistic key sharing

1. INTRODUCTION

Distributed sensor networks have received a lot of attention recently due to their wide application in military as well as civilian operations. Example applications include target tracking, scientific exploration, and monitoring of nuclear power plants. Sensor nodes are typically low-cost, battery powered, and highly resource constrained, and usually collaborate with each other to accomplish their tasks.

Security services such as authentication and key management are critical to secure the communication between sensors 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 sensors to use traditional pairwise key establishment techniques such as public key cryptography and key distribution center (KDC).

Eschenauer and Gligor proposed a probabilistic key predistribution scheme recently for pairwise key establishment [5]. The main idea was to let each sensor node randomly pick a set of keys from a key pool before deployment so any two sensor nodes have a certain probability of sharing at least one common key. Chan et al. further extended this idea and developed two key predistribution techniques: q-composite key predistribution and random pairwise keys scheme [4]. The q-composite key predistribution also uses a key pool but requires two sensors compute a pairwise key from at least q predistributed keys they share. The random pairwise keys scheme randomly picks pairs of sensors 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 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. While the random pairwise keys scheme doesn't suffer from the above security problem, given a memory constraint, the network size is strictly limited by the desired probability that two sensors share a pairwise key and the number of neighbor nodes that a sensor can communicate with.

In this paper, we develop a number of key predistribution techniques to deal with the above problems. In order to facilitate the study of new key distribution techniques, we first develop a general framework for pairwise key establishment based on the polynomial-based key predistribution protocol in [2] and the probabilistic key distribution in [4, 5]. All the previous schemes in [2, 4, 5] are 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 *grid-based* key predistribution scheme. Finally, we present a technique to reduce the computation at sensors so that our schemes can be implemented efficiently.

Our analysis indicates that our new schemes have some nice features when compared with the previous methods. In particu-

CCS'03, October 27–31, 2003, Washington, DC, USA.

lar, 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 for non-compromised sensors to establish secure communication than the previous methods. Moreover, unless the number of compromised sensors sharing a common polynomial exceeds a threshold, compromise of sensors doesn't lead to the disclosure of keys established by non-compromised nodes. Similarly, the grid-based scheme has a number of attractive properties. First, it guarantees that any two sensors can establish a pairwise key when there is no compromised sensors, provided that the sensors can communicate with each other. Second, this scheme is resilient to node compromise. Even if some sensors are compromised, there is still a high probability to establish a pairwise key between non-compromised sensors. Third, a sensor 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 shared keys.

The rest of this paper is organized as follows. Section 2 gives an overview of the polynomial based key predistribution technique. Section 3 presents the general framework for polynomial pool based key predistribution. Sections 4 and 5 describe two instantiations of the framework. Section 6 presents the technique to reduce the computation at sensors. The related work is discussed in Section 7. Section 8 concludes this paper and points out some future research directions. The appendix gives the proof of a Lemma that guarantees the security of the technique presented in Section 6.

2. POLYNOMIAL-BASED KEY PREDISTRIBUTION FOR SENSOR NETWORKS

In this section, we briefly review the polynomial-based key predistribution protocol in [2], which is the basis of our new techniques. The protocol in [2] 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 has a unique ID. For each sensor i, the setup server computes a *polynomial share* of $f(x,y)$, that is, $f(i,y)$. For any two sensor nodes i and j, node i can compute the common 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.

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 [2] 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 [2] 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 no more than t compromised nodes, where the value of t is limited by the memory available in 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 *polynomial pool-based key predistribution*, since a pool of multiple 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 [5] and [4]. The basic idea can be considered as the combination of polynomial-based key predistribution and the key pool idea used in [5, 4]. However, our framework is more general in that it allows different choices to be instantiated within this framework, including those presented in [5, 4] and our later instantiations in Sections 4 and 5.

Intuitively, this general framework uses a pool of randomly generated bivariate polynomials to help establish pairwise keys between sensors. The polynomial pool has two special cases. When the polynomial pool 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 [5, 4].

Pairwise key establishment in this framework is performed in three phases: *setup*, *direct key establishment*, and *path key establishment*. The setup phase is performed to initialize the sensors by distributing polynomial shares to them. After being deployed, if two sensors 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 sensors start path key establishment, trying to establish a pairwise key with the help of other sensors.

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 the 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 polynomial 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.

Phase 2: Direct Key Establishment

A sensor node starts phase 2 if it needs to establish a pairwise key with another node. If both sensors have polynomial shares on the same bivariate polynomial, they can establish the pairwise key directly using the polynomial-based key predistribution discussed in Section 2. Thus, the main issue in this phase is the *polynomial share discovery* problem, which specifies how to find a common bivariate polynomial of which both sensors have polynomial shares. For convenience, we say two sensors have a *secure link* if they can establish a pairwise key through direct key establishment.

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

Predistribution: With this type of techniques, the setup server predistributes certain information to the sensors, so that given the ID of another sensor, a sensor node can determine whether it can establish a pairwise key with the other sensor. A naive method is to let each sensor store the IDs of all the sensors with which it can directly setup a pairwise key. However, this naive method has difficulties dealing with the sensors that join the network on the fly, because the setup server has to inform some existing sensors about the addition of new sensors.

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

Real-time discovery: Intuitively, real-time discovery requires two sensors to discover on the fly whether they both have polynomial shares of a common bivariate polynomial. As one possible way, two sensors 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. For example, using the method in [5], sensor node i may broadcast an encryption list, $\alpha, E_{K_v}(\alpha), v = 1, ..., |\mathcal{F}_i|$, where K_v is a potential pairwise key the other node may have. If node j can correctly decrypt any one of these, it can establish a pairwise key with node i. The drawback of real-time discovery is that it introduces additional communication overhead, which does not appear in the predistribution approaches.

Phase 3: Path Key Establishment

If direct key establishment fails, two sensor nodes will have to start phase 3 to establish a pairwise key with the help of other sensors. For the sake of presentation, we call a sequence of nodes a *path*, or *key path*, since the purpose of such a path is to establish a pairwise key. To establish a pairwise key with node j, a sensor node i needs to find a path between itself and node j such that any two adjacent nodes in the path can establish a pairwise key directly. Then either node i or j initiates a request to establish a pairwise key with the other node through the intermediate nodes along the path. 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 discover a route between themselves so that 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.

Predistribution: Using this type of approach, the setup server predistributes certain information to each sensor node so that given the ID of another sensor, each sensor node can find a key path to the other node directly. The drawback is that an attacker may also take advantage of the predistributed information to attack the network.

Real-time discovery: Real-time discovery techniques have the sensors discover key path on the fly. As one possible way, sensor nodes may take advantage of the pairwise keys established through direct key establishment. To discover a key path to a second sensor, a sensor picks a set of intermediate nodes with which it has established pairwise keys. The source node may send request to all these intermediate nodes. If one of the intermediate nodes can establish a pairwise key with the destination node directly, 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 a source and a destination node. The drawback is that such methods may introduce substantial communication overhead.

4. KEY PREDISTRIBUTION USING RANDOM SUBSET ASSIGNMENT

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

This scheme can be considered as an extension to the basic probabilistic scheme in [5]. Instead of randomly selecting keys from a large key pool and assigning them to sensors, our method randomly chooses polynomials from a polynomial pool and assigns their polynomial shares to sensors. However, our scheme also differs from [5]. In [5], the same key may be shared by multiple sensors. In contrast, in our scheme, there is a unique key between each pair of sensors. If no more than t shares on the same polynomial are disclosed, no pairwise keys constructed using this polynomial between any two non-compromised sensor nodes will be disclosed.

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

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 polynomial shares of these s' polynomials to the sensor node.

Polynomial share discovery: Since the setup server doesn't predistribute enough information to the sensors for polynomial share discovery, sensors 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, a sensor node may broadcast a list of polynomial IDs, or alternatively, broadcast an encryption list $\alpha, E_{K_v}(\alpha), v = 1, ..., |\mathcal{F}_i|$, where K_v is a potential pairwise key the other node may have, as suggested in [5, 4].

Path discovery: If two sensors fail to establish a pairwise key directly, they must start path key establishment phase. During this phase, a source sensor node tries to find another node that can help setup a common key with the destination node. The source node broadcasts a request message, which includes two lists of polynomial IDs (one for the source node and the other for the destination node), to establish a pairwise key. If one of the nodes that receives this request is able to establish a common key with both of the source node and the destination node, it replies with a message that contains two encrypted copies of a randomly generated key: one encrypted by the pairwise key with the source node, and the other encrypted by the pairwise key with the destination node. Both the source and the destination node can then get the new pairwise key from this message. (Note that the intermediate node acts as a KDC in this case.) In practice, we may restrict that a sensor only contact its neighbors within a certain range.

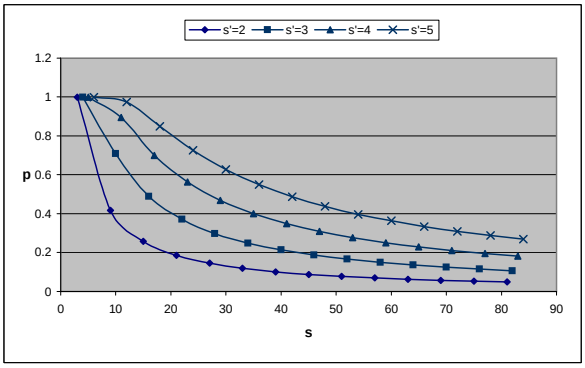
4.1 Analysis

Similar to the analysis in [5], the probability of two sensors sharing the same bivariate polynomial, which is the probability that two sensors can establish a pairwise key *directly*, can be estimated by

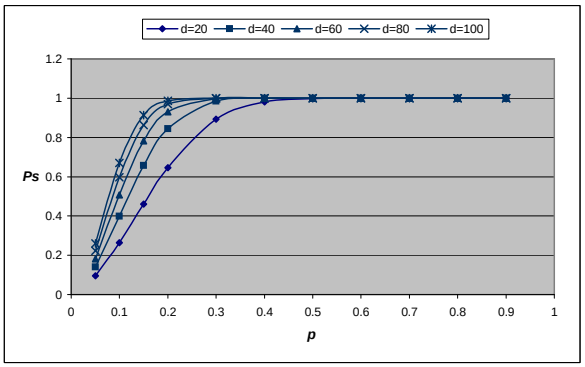
$$p = 1 - \prod_{i=0}^{s'-1} \frac{s - s' - i}{s - i} \tag{1}$$

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 pairwise key directly.

Now let us consider the probability that two sensor nodes can establish a key through both polynomial share discovery and path discovery. Let d denote the average number of neighbor nodes that



(a) The probability p that two sensors 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 sensors share a polynomial

Figure 1: Probabilities about pairwise key establishment

each sensor node can contact. Consider any one of these d neighbor nodes. The probability that it shares a pairwise key with both the source and the destination node is p^2. As long as one of the d nodes can act as an intermediate node, the source and the destination node can establish a common key. It follows that the probability of two sensor nodes establishing a pairwise key (directly or indirectly) is $P_s = 1 - (1-p)(1-p^2)^d$. For example, assuming $p = 0.3$ and $d = 30$, we have $P_s = 1-(1-0.3)(1-0.09)^{30} \approx 0.959$. Figure 1(b) shows the relationship between P_s and the combinations of p and d.

It follows from the security analysis in [2] that an attacker cannot determine non-compromised keys if he/she has compromised no more than t sensors. Now assume an attacker randomly compromises N_c sensors, 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

$$P(i) = \frac{N_c!}{(N_c-i)!i!}(\frac{s'}{s})^i(1-\frac{s'}{s})^{N_c-i}.$$

Thus, the probability of any polynomial being compromised is $P_c = 1 - \sum_{i=0}^{t} P(i)$. Since f is any polynomial in $\mathcal{F}$, the fraction of compromised links between non-compromised sensors can be estimated as P_c. Figure 2 includes the relationship between the fraction of compromised links for non-compromised sensors and the number of compromised nodes for some combinations of s and s'.

If an attacker knows the distribution of polynomials over the sensor nodes, he/she may target at specific sensors in order to compromise the keys derived from a particular polynomial. In this case, the attacker only needs to compromise $t+1$ sensors. However, it is generally more difficult than randomly compromising sensors, 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 sensors. Though effective at improving the security, this method also puts a limit on the maximum number of sensors for a given combination of s and s'. Indeed, given the above constraint, the total number of sensors cannot exceed $\frac{(t+1)\cdot s}{s'}$.

In this scheme, each sensor has to store s' t-degree polynomials over F_q. Thus, the storage overhead is $s'(t+1)\log q$, which is equivalent to storing $s'(t+1)$ keys. During polynomial share discovery, the source node needs to broadcast a list of s' IDs. The communication overhead is mainly due to the transmission of such lists. Once a sensor node determines the polynomial to compute a pairwise key, the computational overhead is mainly due to the evaluation of a t-degree polynomial over F_q.

4.2 Comparison with Previous Schemes

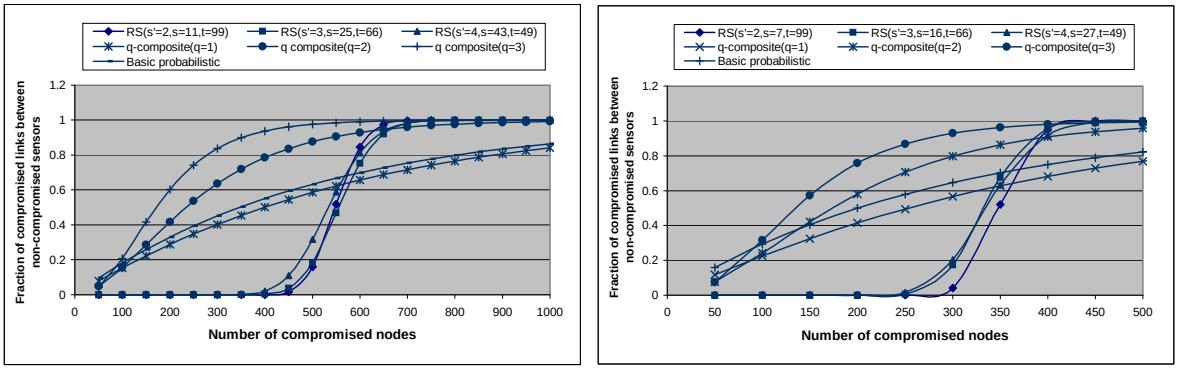
Now let us compare our scheme with the basic probabilistic [5], the q-composite [4], and the random pairwise keys schemes [4].

Figures 2(a) and 2(b) show the security performance of our new scheme, the basic probabilistic scheme [5], and the q-composite scheme [4]. (We will compare our new scheme with the random pairwise keys scheme later.) These figures clearly show that before the number of compromised sensor nodes reaches a certain point, our scheme performs much better than both of the other schemes. When the number of compromised nodes exceeds a certain point, the other schemes have fewer compromised links than ours. Nevertheless, under such circumstances, none of these schemes provide sufficient security due to the large fraction of compromised links (over 60%). Thus, our scheme clearly has advantages over the basic probabilistic scheme [5] and the q-composite scheme [4].

The random pairwise keys scheme does not allow reuse of the same key by multiple pairs of sensors. Thus, compromise of some sensors does not lead to the compromise of links between non-compromised sensors. As we discussed earlier, with a restriction that no polynomial be used more than t times, our scheme can ensure the same property.

Now we compare the performance between our scheme under the above restriction and the random pairwise keys scheme. The maximum number of nodes that our scheme supports can be estimated as $N = \frac{s\times(t+1)}{s'}$. Assuming the storage overhead in each sensor 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 pairwise key directly with a given storage overhead. Figure 3 plots the probability of two sensors sharing a pairwise key directly in terms of the maximum network size for the random pairwise keys scheme [4] and our scheme. We can easily see that our scheme has lower but almost the same performance as the random pairwise keys scheme.

Our scheme has several advantages over the random pairwise keys scheme [4]. In particular, in our scheme, sensors can be added



(a) p=0.33 (b) p=0.5

Figure 2: Fraction of compromised links between non-compromised sensors v.s. number of compromised sensor nodes. RS refers to our scheme. Assume each node has available storage for 200 keys.

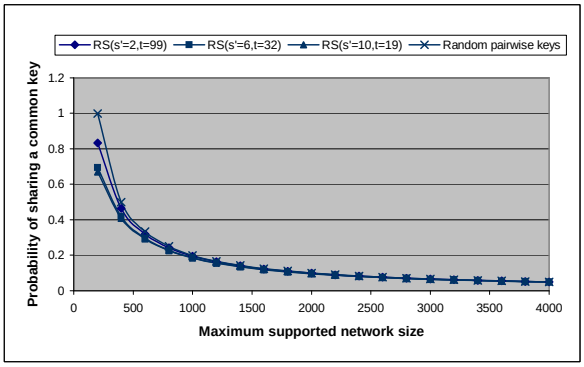


Figure 3: The relationship between the probability of establishing a common key and the maximum supported network size in order to be resilient against node compromise. Assume each node has available storage equivalent to 200 keys.

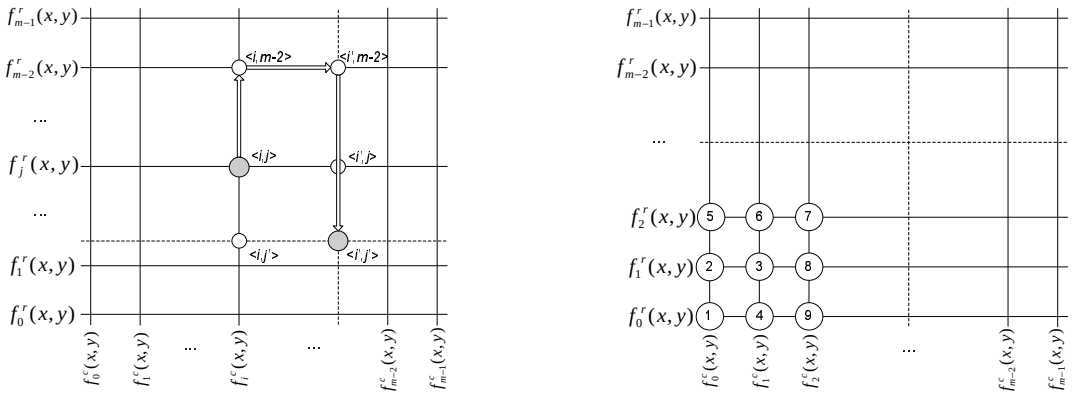
dynamically without having to contact the previously deployed sensors. In contrast, in the random pairwise keys scheme, if it is necessary to dynamically deploy sensors, the setup server has to either reserve space for sensors that may never be deployed, which reduces the probability that two deployed sensors share a common key, or inform some previously deployed sensors of additional pairwise keys. Moreover, given sensor storage constraints, our 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, our scheme would be a more attractive choice than the random pairwise keys scheme in certain applications.

5. GRID-BASED KEY PREDISTRIBUTION

In this section, we give another instantiation of the general framework, which we call *grid-based* key predistribution. This scheme has a number of attractive properties. First, it guarantees that any two sensors can establish a pairwise key when there is no compromised sensors, provided that the sensors can communicate with each other. Second, this scheme is resilient to node compromise. Even if some sensors are compromised, there is still a high probability of establishing a pairwise key between non-compromised sensors. Third, a sensor can directly determine whether it can establish a pairwise 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.

Suppose a sensor network has at most N sensor nodes. The grid-based key predistribution scheme then constructs a $m \times m$ grid with a set of $2m$ polynomials $\{f_i^c(x,y), f_i^r(x,y)\}_{i=0,\ldots,m-1}$, where $m = \lceil \sqrt{N} \rceil$. As shown in Figure 4(a), each row i in the grid is associated with a polynomial $f_i^r(x,y)$, and each column i is associated with a polynomial $f_i^c(x,y)$. The setup server assigns each sensor in the network to a unique intersection in this grid. For the sensor at the coordinate (i,j), the setup server distributes the polynomial shares of $f_i^c(x,y)$ and $f_j^r(x,y)$ to the sensor. As a result, sensor nodes can perform share discovery and path discovery based on this information.

For convenience, we encode the coordinate of a sensor into a single-valued sensor ID. Let $l = \lceil \log_2 m \rceil$. Then any valid column or row coordinate can be represented as an l-bit binary string. We then denote the ID of a sensor as the concatenation of the binary representations of the column and the row coordinates. Syntactically, we represent an ID constructed from the coordinate (i,j) as $\langle i,j \rangle$. For the sake of presentation, we sometimes denote ID i as $\langle c_i, r_i \rangle$, where c_i and r_i are the first and last l bits of i, respectively.



(a) The grid

(b) An example order of node assignment

Figure 4: Grid-based key predistribution

The grid-based key predistribution scheme can be extended from the 2-dimension case to an n-dimension one, or a 2-dimension one with different number of polynomials in each dimension. However, in this paper, we focus on the study of the special 2-dimension scheme, considering the extended schemes as possible future work. The details of the grid-based key predistribution scheme are presented below.

Subset assignment: The setup server randomly generates $2m$ t-degree bivariate polynomials $\mathcal{F} = \{f^c_i(x,y), f^r_i(x,y)\}_{i=0,\ldots,m-1}$ over a finite field F_q. For each sensor, the setup server picks an unoccupied intersection (i,j) in the grid and assigns it to the node. Thus, the ID of this sensor is $ID = \langle i,j \rangle$. The setup server then distributes $\{ID, f^c_i(j,x), f^r_j(i,x)\}$ to this sensor node. To facilitate path discovery, we require that the intersections allocated to sensors are densely selected within a rectangle area in the grid. Figure 4(b) shows a possible order to allocate intersections to the sensors. 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.

Polynomial share discovery: To establish a pairwise key with node j, node i checks whether $c_i = c_j$ or $r_i = r_j$. If $c_i = c_j$, both nodes i and j have polynomial shares of $f^c_{c_i}(x,y)$, and they can use the polynomial-based key predistribution scheme to establish a pairwise key directly. Similarly, if $r_i = r_j$, they both have polynomial shares of $f^r_{r_i}(x,y)$, and can establish a pairwise key accordingly. If neither of these conditions is true, nodes i and j go through path discovery to establish a pairwise key.

Path Discovery: Nodes i and j need to use path discovery if $c_i \neq c_j$ and $r_i \neq r_j$. However, we note that either node $\langle c_i, r_j \rangle$ or $\langle c_j, r_i \rangle$ can establish a pairwise key with both nodes i and j. Indeed, if there is no compromised node, it is guaranteed that there exists at least one node that can be used as an intermediate node between any two sensors due to the node assignment algorithm. For example, in Figure 4(a), both node $\langle i',j \rangle$ and $\langle i,j' \rangle$ can help node $\langle i,j \rangle$ establish a pairwise key with node $\langle i',j' \rangle$. Note that nodes i and j can predetermine the possible intermediate nodes without communicating with others.

In some situations, both of the above intermediate nodes may have been compromised, or are out of communication range. However, there are still alternative key paths. For example, in Figure 4(a), besides node $\langle i',j \rangle$ and $\langle i,j' \rangle$, node $\langle i, m-2 \rangle$ and $\langle i', m-2 \rangle$ can work together to help node $\langle i,j \rangle$ setup a common key with node $\langle i',j' \rangle$. Indeed, there are up to $2(m-2)$ pairs of such nodes in the grid.

In general, we can map the set of non-compromised nodes into a graph, where each vertex in the graph is one of the sensors, and there is an edge between two nodes if these two sensors have polynomial shares of a common polynomial. Discovering a key path between two nodes is equivalent to finding a path in this graph. Nevertheless, in a large sensor network, it is usually not feasible for a sensor to store such a graph and run a path discovery algorithm. Thus, in our scheme, we focus on the key paths that involve two intermediate nodes. Specifically, a sensor node S may use the following algorithm to discover key paths to sensor D that have two intermediate nodes.

1. The source node S determines a set $\mathcal{N}$ of non-compromised nodes that can establish pairwise keys with S directly with a non-compromised polynomial. S randomly picks a set $\mathcal{N}_d$ of d sensor nodes from $\mathcal{N}$. S also generates a random number r, and maintains a counter c with initial value 0.

2. For each node $u \in \mathcal{N}_d$, S increments the counter c and computes $K_c = F(r,c)$, where F is a pseudo random function [6]. Then S sends to u the IDs of S and D, c, and K_c in a message encrypted and authenticated with the pairwise key $K_{S,u}$ between S and u.

3. If a sensor node $u \in \mathcal{N}_d$ receives and authenticates such a message, it knows that node S wants to establish a pairwise key with D. Node u then checks whether the two sensor nodes $\langle c_u, r_D \rangle$ and $\langle c_D, r_u \rangle$ are compromised or not. If u finds a non-compromised node v, u can establish a pairwise key with D through v. Then u sends the IDs of S and D, c, and K_c to v in a message encrypted and authenticated with the pairwise key $K_{u,v}$ between u and v.

4. If v receives the above message and finds that it can establish a pairwise key with D, it further sends the IDs of S and D, c, and K_c to D in a message encrypted and authenticated with the pairwise key $K_{v,D}$ between v and D.

5. Once the destination node D receives a message from such a node v, it knows that the source node S wants to establish a pairwise key $K_{S,D}$ with it. Then it sets $K_{S,D} = K_c$, and

informs S the counter value c. Finally, S and D can use $K_{S,D}$ to secure their communication.

5.1 Analysis

Since each sensor node has two polynomial shares and each bivariate polynomial is shared by about m different sensor nodes, each sensor node can establish a pairwise key with $2(m-1)$ other sensor nodes directly. Thus, among all the other sensors, the percentage of nodes that a node can establish a pairwise key directly is $\frac{2(m-1)}{N-1} \approx \frac{2(m-1)}{m^2-1} = \frac{2}{m+1}$. Moreover, according to the path discovery method, if there is no compromised node, it is guaranteed that any two sensors can establish a pairwise key.

This scheme has reasonable overheads. In terms of storage requirements, each sensor only needs to store 2 t-degree polynomials over F_q. In addition, a sensor need store the IDs of the compromised nodes with which it can establish a pairwise key directly. Thus, the total storage overhead in each sensor is at most $2(t+1)\log q + 2(t+1)l$ bits[1]. In terms of communication overhead, there is none for direct key establishment. When there is an available key path with one intermediate node, there is minor communication overhead, since the sensors know which intermediate node to contact. However, when sensors must discover key paths with two intermediate nodes, there will be a number of unicast messages, depending on how many nodes have been compromised. The computational overhead is essentially the evaluation of one or multiple t-degree polynomials. We will discuss an improvement technique in Section 6.

Now let us turn our attention to the performance of the grid-based key predistribution scheme under attacks. For simplicity, we assume there are $N = m \times m$ sensors in the network.

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

Attacks against A Pair of Sensors

We first look at the attacks against a particular pair of nodes. Certainly, for a particular pairwise key, the attacker can compromise the key if he/she compromises one of the two related sensors. To understand the security of our scheme, we are more interested in how difficult it is to compromise a pairwise key without compromising the related nodes, and how difficult it is to prevent two nodes from establishing a pairwise key.

If nodes u and v can establish a pairwise key directly, the only way to compromise the pairwise key without compromising the related nodes is to compromise the shared polynomial between these two nodes. This requires the attacker to compromise at least $t+1$ sensor nodes. Even if the attacker successfully compromises the polynomial (as well as the pairwise key), the related sensors can still re-establish another pairwise key through path discovery. From the path discovery process, we know that there are still $m-1$ pair of nodes that can help u and v re-establish a pairwise key. To prevent u from establishing a common key with v, the attacker must compromise at least one node in each pair; otherwise, it is still possible to establish a pairwise key between node u and v through multiple rounds of path discovery process. Thus, in this case, the attacker has to compromise $t+1$ nodes to learn the pre-established pairwise key, and at least $t+m$ sensors to prevent u and v from establishing another pairwise key.

Now consider the case in which nodes u and v establish a pairwise key through path key establishment. The attacker may compromise one of the sensors involved in the key path used to establish the pairwise key. If the attacker has the message used to deliver the key, he/she can recover the pairwise key. However, the related sensors can establish a new key with a new round of path key establishment once the compromise is detected. To prevent the sensors from establishing another pairwise key, the attacker has to block all possible key paths between u and v. There are $2m-2$ key paths between u and v that involve one or two intermediate nodes. Besides the key paths with the compromised node, there are at least $2m-3$ paths. To prevent pairwise key establishment, the attacker has to compromise at least one sensor in each path. Thus, in summary, the attacker has to compromise one sensor involved in the path key establishment to compromise the pairwise key, and at least $2m-3$ sensors to prevent u and v from establishing a pairwise key.

Attacks against the Network

Because the adversary knows the subset assignment mechanism, he/she may compromise the bivariate polynomials in $\mathcal{F}$ one after another by compromising selected sensor nodes in order to finally compromise the whole network. Suppose the adversary just compromised l bivariate polynomials in $\mathcal{F}$. There are about ml sensor nodes where at least one of their polynomial shares has been disclosed. Now consider any pair of sensor nodes $u = \langle c_u, r_u \rangle$ and $v = \langle c_v, r_v \rangle$ among the remaining $(m-l)m$ sensor nodes. None of the polynomial shares of these nodes have been compromised. According to the assumption that the adversary just compromised l polynomials, we know that nodes $\langle c_u, r_v \rangle$ and $\langle c_v, r_u \rangle$ have not been compromised, and either of them can help u and v establish a common key. (Indeed, based on our earlier analysis of the attacks against a pair of nodes, even if both nodes have been compromised, there are many other key paths that can help establish a pairwise key between u and v.) Thus, the attacker compromises about $(t+1)l$ sensor nodes ($t+1$ nodes for each bivariate polynomial), but only affects the pairwise key establishment among ml sensor nodes, includes the compromised ones.

As an alternative of the systematic attack, the adversary may randomly compromise sensor nodes to attack the path discovery process, in order to make it more expensive to establish pairwise keys. Assume a fraction of p_c sensor nodes in the network are compromised. Then the probability that exactly k shares on a particular bivariate polynomial have been disclosed is

$$P(k) = \frac{m!}{k!(m-k)!} p_c^k (1-p_c)^{m-k}.$$

The probability of one particular bivariate polynomial being compromised is $P_c = 1 - \sum_{i=0}^{t} P(i)$. Thus, on average, there are $2m \times P_c$ bivariate polynomials being compromised, and about $2m^2 \times P_c$ sensor nodes have one compromised polynomial share.

Consider any pair of non-compromised sensor nodes in the remaining part of the sensor network that have no compromised polynomial share. The probability that the pairwise key between them is compromised is $(1 - \frac{2(m-1)}{N-1}) \times p_c \approx p_c$. These two sensor nodes cannot establish a pairwise key directly, and the sensor node that can help them establish a pairwise key is compromised.

Figure 5(a) shows the relationship between the fraction of compromised links for non-compromised sensors and the number of compromised sensors. We assume each sensor has available storage equivalent to 200 keys. From the figure, we can see that this scheme has a high security guarantee even when a large fraction of

[1]If $t+1$ shares of one bivariate polynomial are compromised, there is no need to remember more compromised sensor IDs, because the polynomial is already compromised. In addition, a sensor node i only needs to remember a half of each ID, because the sensors of concern share either c_i or r_i with node i.

the sensors are compromised. For example, in the case of a sensor network with 20,000 nodes, even if the attacker compromises 50% of the nodes (*i.e.*, 10,000 nodes), only about 0.00131% of the links for non-compromised sensors are compromised. Thus, the majority of the non-compromised nodes are not affected.

Now let us analyze how difficult it is to re-establish a pairwise key between non-compromised sensors when the network is under attack. Assume the attacker randomly compromises a fraction p_c of the sensor nodes. Let us estimate the probability that two non-compromised sensor nodes u and v cannot establish a pairwise key. First, from earlier analysis, we know that the probability that u and v cannot directly establish a pairwise key is $P_{f1} = 1 - \frac{2}{m+1}$. Second, the probability that both $\langle u_c, v_r \rangle$ and $\langle v_c, u_r \rangle$, which are the two sensors that can help u and v establish a common key, are compromised is $P_{f2} = p_c^2$.

Consider the protocol used to discover a key path with two intermediate nodes. Because none of the two polynomials of which u (or v) has shares is compromised, there must be at least $2(m-t-1)$ non-compromised sensors that u (or v) shares a polynomial with. In addition, d is generally a small number, because a sensor usually cannot communicate with too many intermediate sensors due to its limited energy. Thus, it is easy to configure $2(m-t-1) \geq d$. Each non-compromised sensor can then pick at least d sensors to contact during the path discovery. From the path discovery process, the next node that one of the d sensors contacts has probability p_c to be a compromised node. Thus, the probability that this path discovery process fails is $P_{f3} = p_c^d$.

By combining the above three cases, the probability that u cannot establish a pairwise key with v in a single round of path discovery can be estimated by $P_f = P_{f1} \times P_{f2} \times P_{f3} = (1 - \frac{2}{m+1}) p_c^{d+2} = \frac{(m-1)p_c^{d+2}}{m+1}$. Thus, the probability that two remaining sensor nodes can establish a pairwise key is $P_s = 1 - \frac{(m-1)p_c^{d+2}}{m+1} \approx 1 - p_c^{d+2}$. Figure 5(b) shows the relationship between P_s and the fraction of compromised sensor nodes.

5.2 Comparison with Previous Schemes

Let us compare the grid-based key predistribution scheme with the basic probabilistic scheme [5], the q-composite scheme [4], the random pairwise keys scheme [4], and the random subset assignment scheme presented in Section 4.

Assume the network size is $N = 20,000$, and each sensor has the same available storage equivalent to 200 keys). In the grid-based scheme, we have $m = 142$ and $p = 0.014$. The four curves in the right part of Figure 5(a) show the fraction of compromised links as a function of the number of compromised sensors given $p = 0.014$. We can see the basic probabilistic scheme has almost the same performance as the q-composite scheme with $q = 1$. Similar to the comparison in Section 4, the random subset assignment scheme and the grid-based scheme performs much better for less than 14,000 compromised nodes, while none of the schemes can provide sufficient security for more than 14,000 compromised nodes because of the large fraction of compromised links (over 60%).

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 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 5(a) show the fraction of compromised links as a function of the number of compromised sensors for the above three schemes. We can see a small number of compromised nodes reveals a large fraction of secrets in the network for these schemes; however, the fraction of compromised links is much lower in the grid-based scheme for the same number of compromised nodes.

To compare with the random pairwise keys scheme [4], we let $m = t + 1$, so that the grid-based scheme can provide the same degree of perfect security guarantee as the random pairwise keys scheme. Given the same storage overhead of $2(t + 1) = 2m$, we can support a network with m^2 nodes, and the probability that two sensors share a common key directly is $p = \frac{2}{m+1}$. With the same number sensors and storage overhead, the random pairwise keys scheme [4] has $p = \frac{2m}{m^2} = \frac{2}{m}$, which is approximately the same as our scheme.

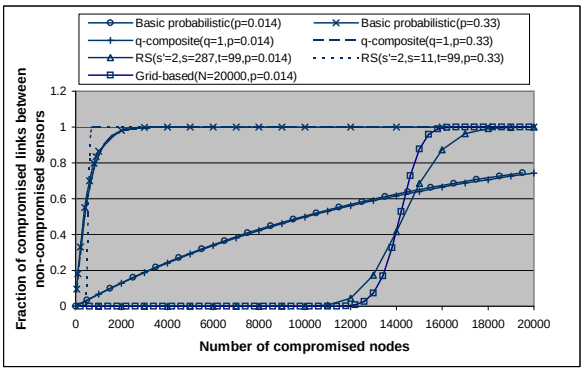
In addition to the above comparisons, the grid-based scheme has some unique properties that the other schemes do not provide. First, when there is no compromised sensors in the network, it is guaranteed that any pair of sensors can establish a pairwise key either directly without communication, or through the help of an intermediate node. 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 sensors in the network, there is still a high probability that two non-compromised sensors can establish a pairwise key. Our earlier analysis indicates that it is very difficult for the adversary to prevent two non-compromised sensors from establishing a pairwise key. In other words, the grid-based scheme is intrusion tolerant in the sense that even if the current pairwise key between two sensors are compromised, as long as these sensors are not compromised, they can re-establish another pairwise key with a high probability. Finally, due to the orderly assignment of grid intersections, this scheme allows optimized deployment of sensors so that the sensors that can establish pairwise key directly are close to each other, thus greatly decreasing the communication overhead in path key establishment.

6. COMPUTATION IN SENSORS

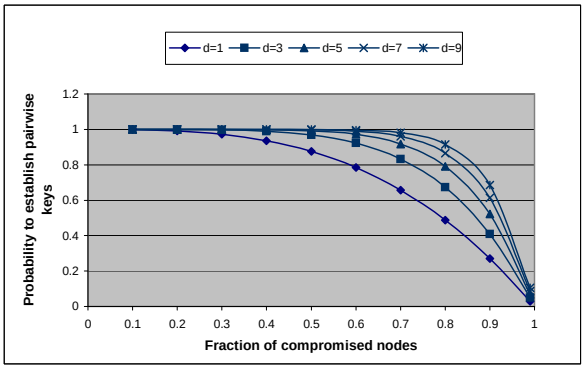
Evaluating a t-degree polynomial is essential in the computation of a pairwise key in our schemes. This requires t modular multiplications and t modular additions in a finite filed F_q, where q is a prime number that is large enough to accommodate a cryptographic key. This implies that q should be at 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.

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 [8]. 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



(a) Fraction of compromised links between non-compromised sensors v.s. number of compromised sensor nodes. Assume each sensor has available storage equivalent to 200 keys.



(b) Probability to establish a pairwise key v.s. the fraction of compromised nodes

Figure 5: Performance of the grid-based key predistribution scheme under attacks

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 $F_{q'}$ [14]. 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 make 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 provide division instruction.

The security of this scheme is guaranteed by Lemma 1.

LEMMA 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 $r = \lceil \frac{n}{l} \rceil$.*

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.

7. RELATED WORK

Our schemes are based on the polynomial-based key predistribution protocol in [2]. The protocol in [2] 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 [5] 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 [4]. This approach allows two sensors 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 are superior to these schemes.

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 [13]. Carman, Kruus, and Matt studied the performance of a number of key management approaches in sensor network on different hardware platform [3]. 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 [15]. Perrig et

al. developed a security architecture for sensor networks, which includes SNEP, a security primitive building block, and μTESLA [12], an adaption of TESLA [10, 11]. In our previous work, we proposed a multi-level key chain method for the initial commitment distribution in μTESLA [9]. Basagni et al. presented a key management scheme to secure the communication by periodically updating the symmetric keys shared by all sensor nodes [1]. However, this scheme assumes a tamper-resistant device to protect the key, which is not always available in sensor networks. Wood and Stankovic identified a number of DOS attacks in sensor networks [16]. 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 [7].

8. CONCLUSIONS 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 [2]. This framework allows study of multiple instantiations of possible pairwise key establishment schemes. As two of the possible instantiations, we developed the key predistribution scheme based on random subset assignment, and the grid-based key predistribution scheme. Our analysis of these schemes demonstrated that both schemes are superior to the existing approaches.

Several directions are worth pursuing in our future research. First, the grid-based scheme can be easily extended to a n-dimensional or hypercube based scheme. We would like to further investigate properties of such extensions and compare them with the existing techniques. Second, we observe that sensor nodes have low mobility in many applications. Thus, it may be desirable to develop location based schemes so that the nodes that can directly establish a pairwise key are arranged to be close to each other.

Acknowledgment We would like to thank Julie M. Starr for proofreading the paper. We would also like to thank the anonymous reviewers for their valuable comments.

9. REFERENCES

[1] S. Basagni, K. Herrin, D. Bruschi, and E. Rosti. Secure pebblenets. In *Proc. of ACM Int'l Symp. on Mobile ad hoc networking and computing*, pages 156–163, 2001.

[2] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. In *Advances in Cryptology – CRYPTO '92, LNCS 740*, pages 471–486, 1993.

[3] D.W. Carman, P.S. Kruus, and B.J.Matt. Constrains and approaches for distributed sensor network security. Technical report, NAI Labs, 2000.

[4] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Research in Security and Privacy*, 2003.

[5] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proc. of the 9th ACM Conf. on Computer and Communications Security*, pages 41–47, November 2002.

[6] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. of the ACM*, 33(4):792–807, October 1986.

[7] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *First IEEE Int'l Workshop on Sensor Network Protocols and Applications*, May 2003.

[8] D.E. Knuth. *The Art of Computer Programming*, volume Vol. 2: Seminumerical Algorithms. Addison-Wesley, third edition, 1997. ISBN: 0-201-89684-2.

[9] D. Liu and P. Ning. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In *Proc. of the 10th Annual Network and Distributed System Security Symposium*, pages 263–276, February 2003.

[10] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient authentication and signing of multicast streams over lossy channels. In *Proc. of IEEE Security and Privacy Symposium*, May 2000.

[11] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient and secure source authentication for multicast. In *Proc. of Network and Distributed System Security Symposium*, February 2001.

[12] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J.D. Tygar. Spins: Security protocols for sensor networks. In *Proc. of Seventh Annual Int'l Conf. on Mobile Computing and Networks*, July 2001.

[13] F. Stajano and R. Anderson. The resurrecting duckling: security issues for ad hoc networks. In *Proc. of 7th Int'l Workshop on Security Protocols*, pages 172–194, 1999.

[14] W. Stallings. *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 2nd edition, 1999.

[15] D. Wong and A. Chan. Efficient and mutually authenticated key exchange for low power computing devices. In *Proc. ASIACRYPT 2001.*, December 2001.

[16] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, October 2002.

APPENDIX

A. PROOF OF LEMMA 1

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 [2], 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

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

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

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