Location-Based Pairwise Key Establishments for Static Sensor Networks

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

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

ABSTRACT

Sensor networks are ideal candidates for applications such as target tracking and environment monitoring. Security in sensor networks is critical when there are potential adversaries. Establishment of pairwise keys is a fundamental security service, which forms the basis of other security services such as authentication and encryption. However, establishing pairwise keys in sensor networks is not a trivial task, particularly due to the resource constraints on sensors. This paper presents several techniques for establishing pairwise keys in static sensor networks. These techniques take advantage of the observation that in static sensor networks, although it is difficult to precisely pinpoint sensors' positions, it is often possible to approximately determine their locations. This paper presents a simple location-aware deployment model, and develops two pairwise key predistribution schemes, *a closest pairwise keys predistribution scheme* and *a location-based pairwise keys scheme using bivariate polynomials*, by taking advantage of sensors' expected locations. The analysis in this paper indicates that these schemes can achieve better performance if such location information is available and that the smaller the deployment error (i.e., the difference between a sensor's actual location and its expected location) is, the better performance they can achieve.

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

Sensor networks are ideal candidates for applications such as military target tracking, home security monitoring, and scientific exploration in dangerous environments. Typically, a sensor network consists of a potentially large number of resource constrained sensors, which are mainly used to collect data (e.g. temperature) from the environment, and a few control nodes, which may have more resources and may be used to control the sensors and/or connect the network to the outside world (e.g. a central data processing server). Sensors usually communicate with each other through wireless communication channels.

Sensor networks may be deployed in hostile environments, especially in military applications. In such situations, the sensors may be captured, and the data/control packets may be intercepted and/or modified. Therefore, security services such as authentication and encryption are essential to maintain the network operations. However, due to the resource constraints on the sensors, many security mechanisms such as public key cryptography are not feasible in sensor networks. Indeed, providing security services in sensor networks is by no means a trivial problem; it has received a lot of attention recently [5, 6, 8, 9, 11, 16, 19].

A fundamental security service is the establishment of a symmetric, pairwise key shared between two sensors, which is the basis of other security services such as encryption and authentication. Several key predistribution techniques have been developed recently to address this problem. Eschenauer and Gligor proposed the basic probabilistic key predistribution, in which each sensor is assigned a random subset of keys from a key pool before the deployment of the network [6]. By doing this, two sensors can have a certain probability to share at least one key. Chan et al. developed the q-composite key predistribution and the random pairwise keys schemes [5]. The q-composite key predistribution scheme is based on the basic probabilistic scheme in [6], but it requires two sensors share at least q predistributed keys to establish a pairwise key. The random pairwise keys scheme predistributes random pairwise keys between a particular sensor and a random subset of other sensors, and has the property that compromised sensors do not lead to the compromise of pairwise keys shared between non-compromised sensors. However, these approaches still have some limitations. For the basic probabilistic and the q-composite key predistribution, a small number of compromised sensors may reveal a large fraction of pairwise keys shared between non-compromised sensors. Though the random pairwise keys scheme provides perfect security against node captures, the maximum supported network size is strictly limited by the storage capacity for pairwise keys and the desired probability to share a key between two sensors [5]. Liu and Ning developed a framework to predistribute pairwise keys using bivariate polynomials and proposed two efficient instantiations, a random subset assignment scheme and a grid-based key predistribution scheme, to establish pairwise keys in sensor networks [10]. Our second scheme in this paper can be considered an instantiation of this framework but can achieve better performance due to the explicit usage of location information.

In this paper, we develop two novel pairwise key predistribution schemes to address the above problems for static sensor networks. These techniques are based on the observation that in static sensor networks, *although it is difficult to precisely pinpoint sensors' positions, it is often possible to approximately determine their locations.* For example, when we use trucks to deploy static sensors, we can usually keep sensors within a certain distance (e.g., 100 yards) from their target locations, though it is difficult to place the sensors in their expected locations precisely. By taking advantage of this observation, our techniques provide better security and performance than the previous techniques.

The contribution of this paper is two-fold. First, we develop a simple location-aware deployment model for static sensor networks, and integrate the location information with the random pairwise keys scheme. The resulting scheme keeps the nice property of the random pairwise keys scheme, that is, the compromise of sensors does not lead to the compromise of pairwise keys shared between non-compromised sensors. However, unlike the random pairwise keys scheme, our scheme does not impose restriction on the network size. Moreover, with the same storage capacity in sensors, our scheme achieves a higher probability to establish pairwise keys than the random pairwise keys scheme, especially when the deployment error is small. Our extension to this basic scheme further allows smaller storage overhead and easier deployment of dynamically added sensors. Second, we develop another location-based key predistribution scheme by combining a polynomial based key predistribution [2] with the location information. This scheme offers further trade-offs between the security against node captures and the probability of establishing pairwise keys between pairs of sensors for a given memory constraint. Our analysis also indicates that this scheme provides higher probability to establish pairwise keys between neighbor sensors and better resistance against node captures than the basic probabilistic scheme [6] and the q-composite scheme [5].

The rest of the paper is organized as follows. Section 2 describes a location-aware deployment model, which will be used in our schemes. Section 3 presents the closest pairwise keys scheme, including a basic version and its extension. Section 4 describes the location-based key predistribution scheme using bivariate polynomials. Section 5 reviews the related work on sensor network security, and Section 6 concludes this paper and points out several future research directions.

2. A LOCATION-AWARE DEPLOYMENT MODEL

In some applications, the sensors may have low mobility, and we may be able to predetermine the location of the sensors to a certain extent. In this case, we can use the sensors' location information to improve the performance of pairwise key predistribution. In this section, we introduce a simple location-aware deployment model for this purpose. We will then develop pairwise key management schemes that can take advantage of the location information in the later sections.

We assume that sensors are deployed in a two dimensional area called the *target field*, and two sensors can communicate with each other if they are within each other's *signal range*. The location of a sensor can be represented by a coordinate in the target field. Each sensor has an *expected location* that can be predicted or predetermined. After the deployment, a sensor is placed at an *actual location* that may be different from its expected location. We call the difference between the expected location and the actual location of a sensor the *deployment error* for the sensor. Thus, this model can be characterized by the following three parameters.

Signal Range d_r: A sensor can receive messages from another sensor if the former is located within the signal range of the latter. We model the signal range of a sensor as a circle centered at its actual location with the radius d_r. For simplicity, we assume the radius d_r defining the signal range is a network-wide parameter, and denote the signal range with d_r. We say two sensors are neighbors if they are physically located within each other's signal range.

Expected Location (L_x, L_y): The expected location (L_x, L_y) of a sensor is a coordinate in the two dimensional target field; it specifies where the sensor is expected to be deployed. Sometimes, a sensor may be expected to be deployed within an area instead of a particular location. In this case, we assume the sensor is expected to be deployed at any location in that area with equal probability.

Deployment Error ϵ: We model the deployment error ϵ with a *probability density function*. The sensor expected to be deployed at (L_x, L_y) may appear at a particular area with certain probability, which is calculated by the integration of probability density function ϵ over this area. In some cases, the sensor may have certain mobility, and appear somewhere near its expected location with a certain probability. The actual location of a sensor at any point in time may also be modeled by the probability density function. We assume the deployment error is also a network-wide parameter.

Obviously, this model can be easily extended to a three dimensional space. However, in this paper, we focus on pairwise key establishments in the two dimensional case. Extending our results to the three dimensional model would be straightforward.

3. CLOSEST PAIRWISE KEYS SCHEME

In this section, we develop a pairwise key management scheme named *closest pairwise keys scheme* to take advantage of the location information. The basic idea is to have each sensor share pairwise keys with c other sensors whose expected locations are closest to the expected location of this sensor, where c is a system parameter determined by the memory constraint. We start with a basic version, which can be considered the combination of the random pairwise keys scheme [5] and the location information, and then present an extended version to further reduce the storage overhead and facilitate dynamic deployment of new sensors.

Throughout this paper, we assume there is a setup server responsible for key predistribution. We assume that the setup server is aware of the network-wide signal range and deployment error, and the expected location of each sensor before deployment. We assume each sensor has a unique, integer-valued ID. We also use a sensor ID to refer to a particular sensor. For example, sensor u implies that the sensor's ID is u. For convenience, we call a pairwise key shared directly between two neighbor nodes as a *direct key*, and a pairwise key established through other intermediate nodes as an *indirect key*.

3.1 The Basic Version

The basic idea of the closest pairwise keys scheme is to *predistribute pairwise keys between pairs of sensors so that two sensors have a predistributed pairwise key if they have a high probability to appear in each other's signal range.* Though reasonable, this idea is difficult to implement, since it is non-trivial to get the probability that two sensors are neighbors. Indeed, this probability depends on the distribution of the deployment error, which is generally not available and may vary in different applications. To simplify the situation, we predistribute pairwise keys between pairs of sensors whose expected locations are close to each other, hoping that the closer the expected locations of two sensors, the more possible that they are physically located in each other's signal range. We will then analyze the probability that two neighbor sensors share a pairwise key using a simple deployment error model. The basic scheme is presented as follows.

Predistribution. Based on the expected locations of the sensors, the setup server predistributes pairwise keys on each sensor to facilitate establishing pairwise keys during the normal operation. Specifically, for each sensor u, the setup server first discovers a set S of c other sensors whose expected locations are closest to the expected location of u. For each sensor v in S, the setup server randomly generates a unique pairwise key $K_{u,v}$ if no pairwise key between u and v has been assigned. The setup server then distributes $(v, K_{u,v})$ and $(u, K_{u,v})$ to sensors u and v, respectively.

Direct Key Establishment. After the deployment of the sensor network, if two sensors u and v want to setup a pairwise key to secure the communication between them, they only need to check whether they have a pre-deployed pairwise key with the other party. This information is obtained from the setup server at the predistribution phase. The algorithm to identify such a common key is trivial, because each pairwise key in a particular sensor was associated with a sensor ID.

Sensor Addition and Revocation. During the lifetime of a sensor network, new sensors may be added to replace the damaged or compromised sensors. To add a new sensor, the setup server performs the above predistribution process for the new sensor, and then informs the deployed sensors chosen for the new sensor the corresponding pairwise keys through secure channels. (Here we assume the communication between each sensor and the setup server is secured with a unique pairwise key shared between the sensor and the setup server.) The setup server may know the actual locations of the deployed sensors. In this case, the setup server may use these locations (instead of their expected locations) to select neighbors for the new sensor.

Sometimes it is necessary to revoke sensors from the secure network possibly due to node compromises. We assume there are other mechanisms to identify sensors to revoke (e.g., through detection of compromised sensors [3, 12, 20]). To revoke a sensor, the other sensors that have a shared pairwise key with the revoked sensor only need to remove the key from their memory.

Indirect Key Establishment. After deployment, if two neighbor sensors u and v do not share a predistributed pairwise key, they may find an intermediate neighbor sensor that shares pairwise keys

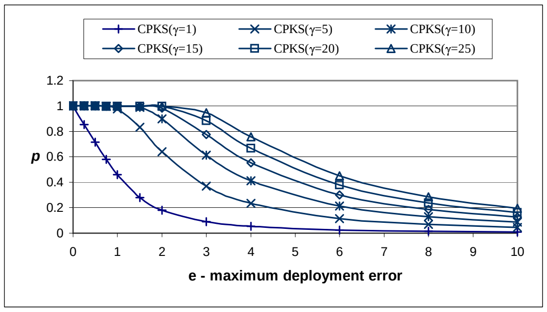


Figure 1: Probability of establishing direct keys between two neighbor nodes given different values of e and γ. CPKS denotes the closest pairwise keys predistribution scheme.

with both of them to help establish a *session key*. Basically, either of these two sensors may broadcast a request message with their IDs. Without loss of generality, we assume u sends this request. Suppose sensor i receives this request, and i shares a pairwise key $k_{u,i}$ with u, and a pairwise key $k_{v,i}$ with v. Sensor i then generates a random session key k and sends a message back to u, which contains $E_{k_{u,i}}(k)$ and $E_{k_{v,i}}(k)$. These are the session key k encrypted with $k_{u,i}$ and $k_{v,i}$, respectively. Upon receiving this reply message, sensor u can get the session key by decrypting $E_{k_{u,i}}(k)$, and inform sensor v by forwarding $E_{k_{v,i}}(k)$ to v. (Note that sensor i acts as a KDC in this case.) Sensor u may receive multiple replies; it may choose any one of them.

Though this scheme looks similar to the previous methods [5, 6], it can achieve better performance if the location information is available. In the following, we show the improvement over previous methods through analysis.

3.1.1 *Analysis*

In this subsection, we analyze the basic closest pairwise keys scheme in detail. We use p to denote the probability of establishing direct keys between neighbor sensors, and use P_c to denote the fraction of compromised direct keys shared between non-compromised sensors.

Probability of Establishing Direct Keys. For simplicity, in later analysis, we assume that the sensors in the network are expected to be evenly distributed in the target field. Thus, if sensor u is one of sensor v's closest c sensors, sensor v is likely to be one of u's closest c sensors. The deployment error for a sensor with expected location (i_x, i_y) is characterized by a probability density function $\epsilon_{i_x,i_y}(x, y)$.

Now consider two sensors u and v that are expected to be deployed at (i_x, i_y) and (j_x, j_y), respectively. The conditional probability that u and v are in each other's signal range given that u is located at (x_1, y_1) can be calculated by

$$p(v_{j_x,j_y}, u_{i_x,i_y}(x_1, y_1)) = \int\int_{d(u,v)\le d_r} \epsilon_{j_x,j_y}(x, y)\mathrm{d}x\mathrm{d}y,$$

where $d(u, v) = \sqrt{(x - x_1)^2 + (y - y_1)^2}$.

Because both u and v are deployed independently, the probability that they are within each other's signal range can be calculated by

$$p(v_{j_x,j_y}, u_{i_x,i_y}) = \int\int \epsilon_{i_x,i_y}(x,y)p(v_{j_x,j_y}, u_{i_x,i_y}(x,y))\mathrm{d}x\mathrm{d}y.$$

Assume there are m nodes on average in each sensor's signal range. The density of the network can be estimated by $D = \frac{m}{\pi d_r^2}$, where d_r is the radius of the signal range. Thus, on average, each sensor will get pre-deployed pairwise keys with the sensors that are no more than d' away from it, where $d' = \sqrt{\gamma} \times d_r$, and $\gamma = \frac{c}{m}$. We call $\gamma = \frac{c}{m}$ the *capacity density ratio*. For any v having a pre-deployed pairwise key with u, the probability that v falls into u's signal range is

$$p(u_{i_x,i_y}) = \int\int_{(x-i_x)^2+(y-i_y)^2 \le d'^2} \frac{p(v_{x,y}, u_{i_x,i_y})}{\pi d'^2}\mathrm{d}x\mathrm{d}y.$$

Among the sensors that have predistributed pairwise keys with sensor u, the average number of sensors that fall into its signal range can be estimated by $c \times p(u_{i_x,i_y})$. Thus, the probability of establishing a common key between any two neighbor sensors can be estimated by

$$p = \frac{c \times p(u_{i_x,i_y})}{m} = \gamma \times p(u_{i_x,i_y}).$$

The above analysis can be applied to any deployment error model characterized by a probability density function. In this paper, we evaluate the performance of our scheme under a simple one in which each sensor may randomly appear anywhere at a distance of no more than e away from the expected location. We call e the *maximum deployment error*. Thus, the deployment error for a sensor with expected location (L_x, L_y) can be expressed by

$$\epsilon_{L_x,L_y}(x,y) = \begin{cases} \frac{1}{\pi e^2}, & (x-L_x)^2 + (y-L_y)^2 \le e^2; \\ 0, & otherwise. \end{cases}$$

We use the radius of signal range, d_r, as the basic unit of distance measurement ($d_r = 1$). Figure 3.1.1 shows the probability of establishing direct keys between neighbor sensors for different values of e and γ. We can see that this probability is not only affected by the deployment error, but also by the capacity density ratio γ. In general, the increase of γ will increase the probability p given certain deployment error. However, this probability decreases when the maximum deployment error increases. In practice, we may expect to see better performance than in Figure 3.1.1, since the probability of having a smaller deployment error is typically higher than the probability of having a larger one.

Security Against Node Captures. From the scheme it is easy to see that each predistributed pairwise key between two sensors is randomly generated. Thus, no matter how many sensors are compromised, the direct keys between non-compromised sensors are still secure. Once a sensor is compromised, the session keys this sensor helps establish may be compromised. For example, an attacker may have saved a copy of the reply message from this sensor, and thus is able to decrypt the session key once she gets the predistributed pairwise keys. Thus, if either of the source and destination sensors notices that the intermediate sensor is compromised, it should remove the corresponding predistributed pairwise key, and initiate a request to establish a new session key. The delay in detecting compromised sensors still poses a threat. One way to mitigate this threat is to derive the session key by combining (e.g., XOR) the keys generated by at least q ($q > 1$) different intermediate sensors, adopting the idea in the q-composite scheme [5].

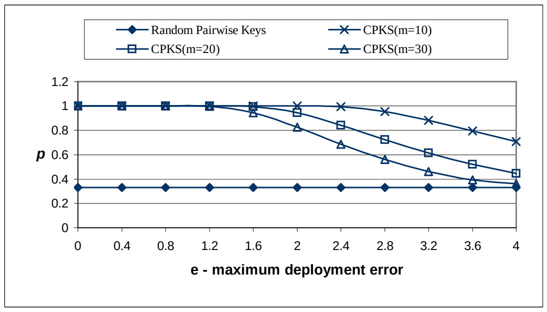


Figure 2: Probability of establishing direct keys in random pairwise keys scheme and the closest pairwise keys scheme for different m and e given $c = 200$ and $N = 600$.

Overhead. Ideally, each sensor stores c pairwise keys. However, this does not necessarily happen because of the asymmetry in sensors' locations. Consider a pair of sensors u and v. In the predistribution step, v is one of u's closest c sensors; however, u is not necessarily one of v's closest c sensors. In this case, v has to store the pairwise key between u and v in addition to its own predistributed c pairwise keys. Thus, the storage overhead in each sensor comes from two parts: One consists of the pairwise keys generated for itself, and the other consists of the pairwise keys generated for other sensors. Hence, each sensor has to store at least c keys and c sensor IDs. The actual number of pairwise keys stored in a particular sensor may be much larger than c. Nevertheless, if the sensors are approximately evenly distributed in the target field, it is very likely that if sensor u is among sensor v's closest c sensors, then v is among u's closest c sensors.

To establish a common key with a given neighbor node, a sensor only needs to check whether it has a predistributed pairwise key with the given node (because each pairwise key is associated with a sensor ID). Thus, there is no communication and computation overhead during a direct key establishment. Establishment of an indirect session key requires one broadcast request message, and potentially a number of unicast reply messages. However, from Figure 3.1.1, we can see this is highly unlikely given reasonable capacity density ratio and maximum deployment error.

Comparison with previous methods. Let's first compare our scheme with the random pairwise keys scheme in [5]. Our basic scheme can be considered an extension to the random pairwise keys scheme. These two schemes have some common properties. In both schemes, compromise of sensors does not lead to the compromise of direct keys shared between non-compromised sensors. However, our scheme further takes advantage of the location information, and thus is able to achieve better performance than the random pairwise keys scheme. First, the random pairwise keys scheme has a restriction on the number of sensors that can be in the same network, while our scheme has no limitation on the network size. Second, given the same storage capacity c for pairwise keys and the total number N of sensors, the probability of establishing direct keys in our scheme is always better than the random pairwise keys scheme. This is illustrated in Figure 2, which compares the probability of establishing direct keys in both schemes for different

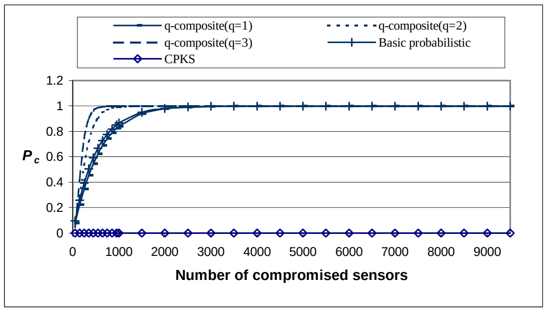


Figure 3: Fraction of compromised pairwise keys between non-compromised sensors v.s. Number of compromised sensors.

m and e given that $c = 200$ and $N = 600$. Figure 2 shows the probability p of establishing direct keys is improved significantly in our scheme, especially when e is less than two times of the signal range. When the maximum deployment error e increases, this probability gradually decreases, and eventually merges into the line for the random pairwise keys scheme.

Now let's compare our basic scheme with the basic probabilistic scheme [6] and the q-composite scheme [5]. As discussed earlier, our scheme proposed has a high probability to establish direct keys between neighbor sensors given reasonable capacity density ratio γ and maximum deployment errors. At the same time, our scheme does not put any limitation on the network size. In addition, our scheme is much more secure than these two schemes when there are compromised nodes. Figure 3 shows that given the same storage overhead and the same value of p, our scheme does not lead to compromise of direct keys belonging to non-compromised sensors (as discussed earlier), while in the other two schemes [5, 6], the direct keys shared between non-compromised sensors are compromised quickly when the number of compromised sensors increases.

3.2 The Extended Version

The pairwise keys predistribution technique described above has two limitations. First, if the sensors are not evenly distributed in the target field, it is possible for a sensor to have a large number of neighbor sensors that are not among the closest c sensors of u, but consider u as among their closest c sensors. As a result, this sensor has to store a lot of pairwise keys generated by the setup server. Second, to add a new sensor after deploying the sensor network, the setup server has to inform a number of existing sensors in the network about the addition of the new sensor, which may introduces a lot of communication overhead.

In this subsection, we propose an alternative way to predistribute the secret information so that (1) the storage overhead in each sensor is small and fixed no matter how the sensors are deployed, and (2) no extra communication overhead is introduced during the addition of new sensors. The technique is based on a pseudo random function (PRF) [7] and a master key shared between each sensor and the setup server.

In the following, we only describe the predistribution, direct key establishment, and sensor addition and revocation procedures. The other parts are the same as in the basic scheme.

Predistribution. For each sensor u, the setup server first randomly generates a master key K_u. The setup server also determines a set S of c other sensors whose expected locations are closest to that of u. The setup server then distributes to sensor u a set of pairwise keys (together with the IDs) with those selected sensors, which the setup server generates in the following way: For each $v \in S$, the setup server generates a pseudo random number $k_{u,v} = PRF_{K_v}(u)$ as the pairwise key shared between u and v, where K_v is the master key for v. As a result, for each $v \in S$, sensor u stores the pairwise key $k_{u,v}$, while sensor v can compute the same key with its master key and the ID of sensor u. We call v a *master sensor* of u if the direct key $k_{u,v}$ shared between them is derived by $k_{u,v} = PRF_{K_v}(u)$. Accordingly, we call u a *slave sensor* of v if v is a master sensor of u.

Direct Key Establishment. The direct key establishment stage is similar to the basic scheme. The only difference is that one of two sensors has a predistributed pairwise key and the other only needs to compute the key using its master key and the ID of the other party. For example, if u finds that it has the pre-deployed pairwise key $PRF_{K_v}(u)$ with v, it then notifies sensor v that it has such a key. Sensor node v only needs to compute $PRF_{K_v}(u)$ by performing a single pseudo random function.

Sensor Addition and Revocation. To add a new sensor u, the setup server selects c sensors closest to the expected location of u. For each of these c sensors, the setup server retrieves v's master key K_v and computes $k_{u,v} = PRF_{K_v}(u)$, and then distributes v and $k_{u,v}$ to u. Revoking a sensor is a little more complex than in the basic scheme. To revoke sensor v, all its slave sensors need to remove the corresponding keys from their memory. However, v's master sensors have to remember v's ID in order to avoid establishing a direct key with v later.

In this extension, each sensor needs to store a master key which is shared with the setup server and c pre-deployed pairwise keys. Thus, the storage overhead for keys in each sensor is at most $c+1$. To establish a pairwise key, one of them can initiate a request by notifying the other party that it has the pre-deployed pairwise key. (Note that this message only indicates the existence of such key and nothing about what this key looks like is disclosed on the communication channel.) Once the other party receives such message, it can immediately compute the pairwise key by performing one efficient PRF operation. Thus, the communication overhead in the above scheme involves only one short request message and the computation overhead only involves one efficient PRF operation.

Based on the security of PRF [7], if a sensor's master key is not disclosed, no matter how many pairwise keys generated with this master key are disclosed, it is still computationally infeasible for an attacker to recover the master key and the non-disclosed pairwise keys generated with different IDs. Thus, the compromise of sensors does not lead to the compromise of the direct keys shared between non-compromised sensors.

The extended scheme introduces some additional overhead by requiring master sensors to remember the IDs of their revoked slaves. We consider this an acceptable overhead due to the following reasons. First, the storage overhead for a sensor ID is much smaller than that for one cryptographic key. Second, in normal situations when authentication of the revocation information is ensured, the number of revoked slave sensors is usually less than m, the average number of sensors in each sensor's signal range. One may argue that if the authentication of revocation information can be bypassed, an attacker may convince a sensor to store many sensor IDs in order to exhaust its memory. However, in this case, the sensor can be convinced to do almost anything, and should be considered compromised.

4. LOCATION-BASED KEY PREDISTRIBUTION USING BIVARIATE POLYNOMIALS

The schemes presented earlier still has some limitations. In particular, given the constraints on the storage capacity, sensor density, signal range and deployment error, the probability of establishing direct keys is fixed. For a particular sensor network, it is not convenient to adjust the last three parameters. Thus, one has to increase the storage capacity for pairwise keys to increase the probability of establishing direct keys. This may not be a feasible solution in certain sensor networks given the memory constraints on sensors.

Here we develop a key predistribution scheme to address the above limitation by using bivariate polynomials and the location information. The resulting technique allows trade-offs between the security against node captures and the probability of establishing direct keys with a given memory constraint. Moreover, it does not require the setup server be aware of the global network topology, making the deployment easier.

In the following, we first review a polynomial-based key predistribution technique, then present the location-based key predistribution scheme, and finally analyze the security and performance of this scheme.

4.1 Key Predistribution with Bivariate Polynomials

Our scheme is based on the polynomial-based key predistribution technique proposed in [2], which was developed for group key predistribution. Though using the technique in [2] for group key predistribution is generally not practical because of its overhead, its special case for pairwise keys is feasible in sensor networks. For simplicity, we only discuss the special case of pairwise key establishment next.

To predistribute pairwise keys, the setup server randomly generates a bivariate t-degree polynomial $f(x, y)$ 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 $f(x, y)$, $f(i, y)$. 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. 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 collusion of no more than t compromised sensor nodes knows nothing about the direct key between any two non-compromised nodes.

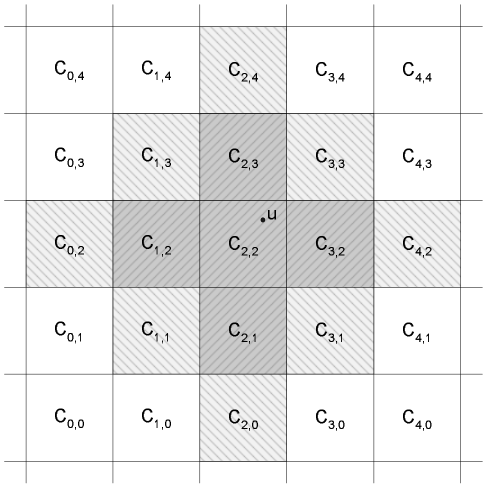


Figure 4: Partition of a target field

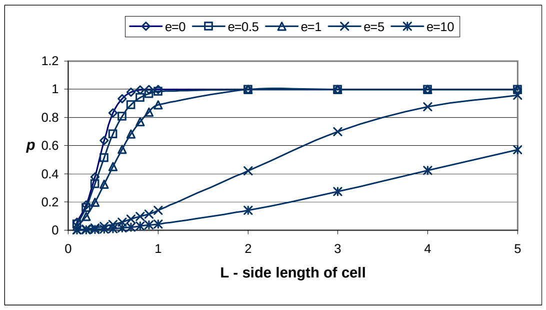
4.2 Our Scheme

The polynomial-based key predistribution scheme discussed above can only tolerate no more than t compromised nodes, where the value of t is limited by the storage capacity for pairwise keys in a sensor. Indeed, the larger a sensor network is, the more likely that an adversary compromises more than t sensors and then the entire network. However, the polynomial-based key predistribution introduces an interesting threshold property, that is, an attacker has to compromise more than a certain number of sensors in order to compromise the secret. Our scheme in this section aims to integrate this property with location information.

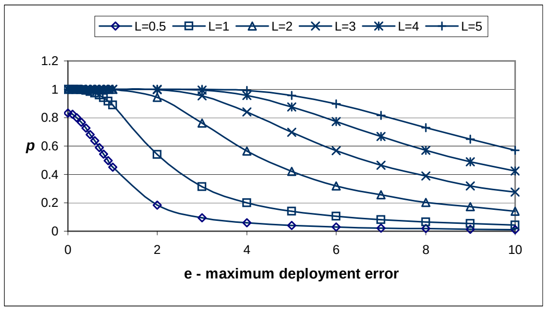
The main idea of the proposed technique is to combine the closest pairwise keys scheme with the polynomial-based key predistribution technique. Specifically, we partition the target field into small areas called *cells*, each of which is associated with a unique random bivariate polynomial. Then, instead of assigning each sensor the pairwise keys for the closest sensors, we distribute to each sensor a set of polynomial shares that belong to the cells closest to the one that this sensor is expected to locate in. For simplicity, we assume the target field is a rectangle area that can be partitioned into equal-sized squares.

Predistribution. Partitions the target field into equal sized squares $\{C_{i_c,i_r}\}_{i_c=0,1,\ldots,C-1, i_r=0,1,\ldots,R-1}$, each of which is a *cell* with the coordinate (i_c, i_r) denoting row i_r and column i_c. For convenience, we use $s = R \times C$ to denote the total number of cells. The setup server randomly generates s bivariate t-degree polynomials $\{f_{i_c,i_r}(x, y)\}_{i_c=0,1,\ldots,C-1, i_r=0,1,\ldots,R-1}$, and assigns $f_{i_c,i_r}(x, y)$ to cell C_{i_c,i_r}. Figure 4 shows an example partition of a target field.

For each sensor, the setup server first determines its *home cell*, in which the sensor is expected to locate. The setup server then discovers four cells adjacent to the sensor's home cell. Finally, the setup server distributes to the sensor its home cell coordinate and the polynomial shares of the polynomials for its home cell and the four selected cells. For example, in Figure 4, sensor u is expected to be deployed in cell $C_{2,2}$. Obviously, cell $C_{2,2}$ is its home cell, and cell $C_{2,1}$, $C_{1,2}$, $C_{2,3}$ and $C_{3,2}$ are the four cells adjacent to its home cell. Thus, the setup server gives this sensor the coordinate $(2, 2)$ and the polynomial shares $f_{2,2}(u, y)$, $f_{2,1}(u, y)$, $f_{1,2}(u, y)$,



(a)



(b)

Figure 5: Probability of establishing direct keys between two neighbor nodes given different cell side length L and maximum deployment error e

$f_{2,3}(u,y)$, and $f_{3,2}(u,y)$.

Direct Key Establishment. After deployment, if two sensors want to setup a pairwise key, they first need to identify a shared bivariate polynomial. If they can find at least one such polynomial, a common pairwise key can be established directly using the basic polynomial-based key predistribution presented in Section 4.1. A simple way is to let one of them (called *source node*) disclose its home cell coordinate to the other node (called *destination node*). From the coordinate of the home cell of the source node, the destination node can immediately determine the set of polynomial shares the source node has. To protect this coordinate information, the source node may challenge the destination node to solve puzzles. For example, using the method in [6], the source node may send an encryption list, $\alpha, E_{K_v}(\alpha)$, $v = 1, ..., 5$, where K_v is a potential pairwise key the other node may have. If the destination node can correctly decrypt one of them, it can establish a pairwise key with the the requesting node and thus send a short reply message to identify the common shared key.

Sensor Addition and Revocation. To add a new sensor, the setup server only needs to predistribute the related polynomial shares and the home cell coordinate to the new sensor, in the same way as in the predistribution phase. The revocation method is also straightforward. Each sensor only needs to remember the IDs of the compromised sensors that shares at least one common bivariate polynomial with itself. Thus, in addition to the polynomial shares, the sensor also needs to store a number of compromised sensor IDs. If more than t sensors that share the same bivariate polynomial are compromised, a non-compromised sensor that has a share of this polynomial simply remove the corresponding share and all the related compromised sensor IDs from its memory.

Indirect Key Establishment. Same as the closest pairwise key scheme.

4.3 Analysis

In this subsection, we give a detailed analysis of the above key predistribution scheme.

Probability of Establishing Direct Keys. Consider two sensors u and v that are expected to be deployed in cell C_{i_c,i_r} and C_{j_c,j_r}, respectively. The conditional probability of being in each other's signal range given that (1) u is expected to be at (i_x, i_y) but is actually deployed at (x_1, y_1), and (2) v is expected to be deployed at (j_x, j_y) can be calculated by

$$p(v_{j_x,j_y}, u_{i_x,i_y}(x_1,y_1)) = \int\int_{d(u,v)\le d_r} \epsilon_{j_x,j_y}(x,y)\mathrm{d}x\mathrm{d}y,$$

where $d(u,v) = \sqrt{(x-x_1)^2 + (y-y_1)^2}$.

Since u and v are deployed independently, the conditional probability of being in each other's signal range given that u and v are expected to be deployed at (i_x, i_y) and (j_x, j_y), respectively, can be calculated by

$$p(v_{j_x,j_y}, u_{i_x,i_y}) = \int\int \epsilon_{i_x,i_y}(x,y)p(v_{j_x,j_y}, u_{i_x,i_y}(x,y))\mathrm{d}x\mathrm{d}y.$$

To simplify our analysis, we assume that a sensor is expected to locate randomly in its home cell. In other words, if sensor v is expected to be in cell C_{j_c,j_r}, then the probability density function for the expected location of v is $\frac{1}{L^2}$ for any location in the cell, and 0 otherwise. Therefore, the conditional probability that u and v are in each other's signal range given that u is expected to be deployed at location (i_x, i_y) and v is expected to be deployed in cell C_{j_x,j_y} can be calculated by

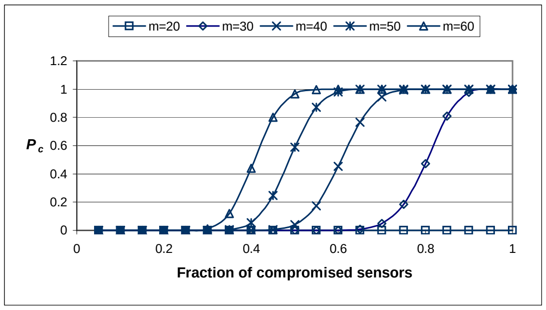
$$p(C_{j_c,j_r}, u_{i_x,i_y}) = \int\int_{C_{j_c,j_r}} \frac{p(v_{x,y}, u_{i_x,i_y})}{L^2}\mathrm{d}x\mathrm{d}y.$$

Hence, the probability of being able to establish a common key between u and v given that u and v's home cells C_{i_c,i_r} and C_{j_c,j_r} can be estimated by

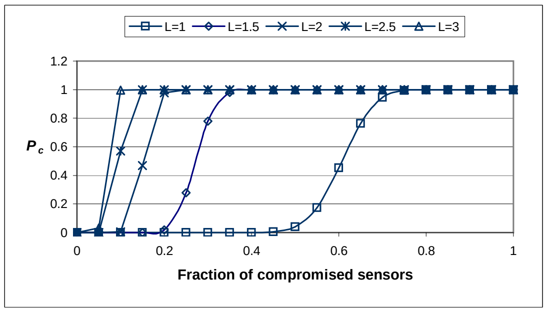
$$p(C_{j_c,j_r}, C_{i_c,i_r}) = \int\int_{C_{i_c,i_r}} \frac{p(C_{j_c,j_r}, u_{x,y})}{L^2}\mathrm{d}x\mathrm{d}y.$$

Assume that on average, N_{cell} sensors are expected to be deployed in each cell. Thus, among all the sensors with home cell C_{j_c,j_r}, the average number of sensors that a sensor u can directly communicate with can be estimated by $N_{cell} \times p(C_{j_c,j_r}, C_{i_c,i_r})$. Therefore, overall, the average number of sensors that u can directly communicate with can be estimated by

$$n_u = N_{cell} \cdot \sum_{\forall j_c,j_r} p(C_{j_c,j_r}, C_{i_c,i_r}).$$



(a) L=1



(b) m=40

Figure 6: Fraction of compromised direct keys between non-compromised sensors v.s. Fraction of compromised sensors. Assume each node has available storage equivalent to 200 cryptographic keys.

Let $\mathcal{S}_{i_c,i_r}$ denotes the set of home cells of the sensors that share at least one common polynomial with a sensor whose home cell is C_{i_c,i_r}. According to our predistribution procedure, there are 13 such cells in each $\mathcal{S}_{i_c,i_r}$. As an example, Figure 4 shows $\mathcal{S}_{2,2}$, which consists of all the shaded cells. Thus, the average number of sensors that can establish a common key with u directly can be estimated by

$$n_u^s = N_{cell} \cdot \sum_{C_{j_c,j_r} \in \mathcal{S}_{i_c,i_r}} p(C_{j_c,j_r}, C_{i_c,i_r}).$$

Thus, the probability of establishing a common key directly between u and its neighbor nodes, which is also the probability of establishing a common key directly between any two neighbor nodes, can be estimated by

$$p = p(u) = \frac{n_u^s}{n_u} = \frac{\sum_{C_{j_c,j_r} \in \mathcal{S}_{i_c,i_r}} p(C_{j_c,j_r}, C_{i_c,i_r})}{\sum_{\forall j_c,j_r} p(C_{j_c,j_r}, C_{i_c,i_r})}.$$

We use the simple deployment error model described in the previous section to evaluate the performance, with signal range d_r as the basic unit for distance measurement ($d_r = 1$). Figure 5 shows the probability of establishing direct keys for different combinations of cell side length L and maximum deployment error e. Obviously, the probability of establishing direct keys increases with the cell side length L and decreases with the maximum deployment error e.

In general, the larger L is, the higher the probability of establishing a direct key between two neighbor nodes. However, the larger cell side length also leads to a larger number of sensors sharing the same bivariate polynomial, which in turn degrades the security performance. Thus, we have to find the minimum value of L to meet the other constraints so that we can maximize the security performance. Figure 5 provides a guideline to determine the minimum value of L given the other constraints.

Security against Node Captures. According to the result of the polynomial-based key predistribution, as long as no more than t polynomial shares of a bivariate polynomial are disclosed, an attacker knows nothing about the non-compromised pairwise keys established through this polynomial. Thus, the security of our scheme depends on the average number of sensors sharing the same polynomial, which is equivalent to the number of sensors that are expected to be located in the corresponding cell and its four adjacent cells.

As discussed in Section 3, the density of the sensors in the network can be estimated by $D = \frac{m}{\pi d_r^2}$. The average number of sensors that are expected to be located in a cell is $\frac{m*L^2}{\pi d_r^2}$. Thus, the average number of sensors that share the polynomial of a particular cell can be estimated by $N_s = \frac{5*m*L^2}{\pi d_r^2}$. Using the signal range as the basic unit of distance measurement, we have $N_s = \frac{5*m*L^2}{\pi}$.

We consider two attacks in this subsection. One is a *localized attack*, which targets at the sensors in a particular area in order to compromise the communication security in this area. The other is a *random attack*, which randomly selects sensors to compromise.

In a localized attack, the attacker must compromise more than t out of N_s sensors in order to compromise the keys between non-compromised sensors in that area. In addition, the compromise of a particular area does not affect the keys in any other area because all bivariate polynomials are chosen randomly and independently.

Consider a random attack. We assume a fraction p_c of sensors in the network have been compromised by an attacker. This means that each sensor has the probability of p_c being compromised. Thus, among N_s sensors that have polynomial shares of a particular cell, the probability that exactly i sensors have been compromised can be estimated by

$$P_c(i) = \frac{N_s!}{(N_s - i)!i!} p_c^i (1 - p_c)^{N_s - i}.$$

Thus, the probability that the bivariate polynomial assigned to this cell is compromised is $P_c = 1 - \sum_{i=0}^{t} P_c(i)$. For any pairwise key established directly between non-compromised sensors, the probability that it is compromised is the same as P_c. Figure 6 includes the relationship between the fraction of compromised direct keys for non-compromised sensors and the fraction of compromised nodes under different combination m and L given the storage capacity that is equivalent to 200 cryptographic keys ($t = 39$). An interesting result is that regardless of the total number of sensors in the network, the less the density of the sensor network, the higher the security guarantee it can provide.

Comparison. Now let's compare our scheme in this section with previous methods (basic probabilistic scheme [6], the q-composite scheme [5], the random pairwise keys scheme [5], and the closest pairwise keys scheme). Evaluation of those schemes requires the network size. To be fair, we use the following method to estimate the network size in our scheme. Assume on average, there are m sensors that fall into each sensor's signal range. Based on the analysis in [5], we estimate the total number of sensors in the network is $N = 2^{m*p}$ to make sure the network is fully connected at a high probability, where p is the probability of establishing direct key between two neighbor sensors.

Let's first compare our new scheme with the basic probabilistic scheme [6] and the q-composite scheme [5]. Figure 7(a) compares the number of compromised keys shared between non-compromised sensors given the same p, m, and storage overhead. We can see that our scheme significantly better than the other two schemes. It also shows that the more precise the sensor deployment is, the higher security it can guarantee.

We then compare our new scheme in this section with the random pairwise keys scheme [5]. By limiting the number of sensors sharing the same bivariate polynomial, our proposed scheme can be modified to provide perfect security against node captures. Then, we have $N_s = \frac{5*m*L^2}{\pi} \leq (t+1)$. From the previous result, we know that the value of p only depends on L and e. Thus, given the same probability of establishing direct keys between sensors, our proposed scheme has no limit on the total number of sensors it can support. However, the random pairwise key scheme can only support at most $\frac{c}{p}$ sensors, where c is the number of cryptographic keys a sensor stores [5]. Thus, our new scheme can achieve better performance when the location information is available.

Now we compare our new scheme proposed in this section with the closest pairwise keys scheme in Section 3. For the closest pairwise key predistribution, given a fixed storage capacity c, signal range d_r, sensor density D, and the maximum deployment error e, the probability of establishing direct keys between sensors is fixed. However, for our new scheme proposed in this section, given the above constraints, it can still achieve arbitrary high probability to establish direct keys between sensors by increasing the cell side length L as shown in Figure 5. For example, in the closest pairwise keys scheme, if $\gamma = 5$, $e = 3$, the probability of having a common pairwise key between two neighbor nodes is 0.4. Thus, if on average, there are 20 other sensors within each sensor's signal range, the closest pairwise keys scheme can only establish direct keys for 8 nodes. Then, to make sure the overall network is fully connected, it can only support $2^8 = 256$ sensor nodes. In contrast, our new scheme allows to increase cell side length to achieve the required p and still provide certain degree of security.

An advantage of the closest pairwise key scheme is that compromise of sensors does not lead to the compromise of direct keys shared between non-compromised sensors. We call this property as *perfect resistance against node captures*. By having $N_s \leq (t+1)$, the location-based scheme proposed in this section can also provide this security property. To further compare these two schemes under this condition, Figure 7(b) shows the probabilities of establishing direct keys with storage capacity which is equivalent to 200 cryptographic keys and different sensor densities and maximum deployment errors. We can see that although the closest pairwise keys scheme has a higher probability to establish direct keys between neighbor sensors, our new scheme is not significantly worse, especially for a large deployment error. Considering the flexibility to trade-off security and performance in the new scheme in this section, we can conclude that the location-based key predistribution scheme using bivariate polynomials is a desirable alternative for the closest pairwise keys scheme.

Overhead. Each sensor needs to store the coordinate of its home cell and the polynomial shares of five cells. The storage overhead for the coordinate of its home cell is negligible. Thus, each sensor has to allocate $5(t+1)\log q$ memory space to store the secret. When there are compromised sensors, each non-compromised sensor also needs to store the IDs of the compromised sensors with which it shares at least one polynomial. However, for each of the 5 polynomials, a non-compromised sensor only needs to store up to t IDs; it can remove the corresponding polynomial share and all the related IDs if the number of compromised sensors with which it shares the polynomial exceeds t.

To establish a common key between two neighbor nodes, one of them initiates a request by disclosing its home cell coordinate or giving an encryption list as in [6]. Once the other party receives such a message, it can immediately determine the common pairwise key and reply a message to identify the corresponding key. Thus, the communication overhead includes two messages.

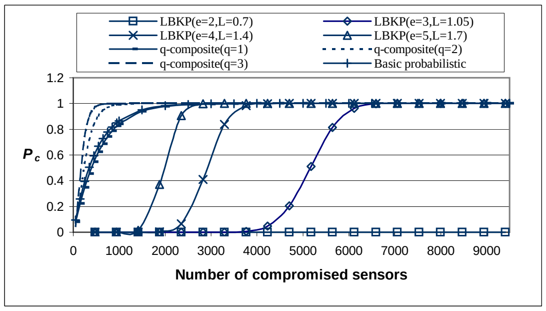
To compute the common key with a given sensor, each sensor node needs to evaluate a t-degree polynomial. Thus, the computational cost in each sensor mainly comes from the evaluation of this polynomial, which requires t modular multiplication and t modular addition. Note that this is different from the modular operations in a public key cryptosystem, which involves operations modulo a very large integer (e.g., 1024 bits). In our scheme, the modular just needs to be big enough to accommodate a symmetric key (e.g., 65 bits).

5. RELATED WORK

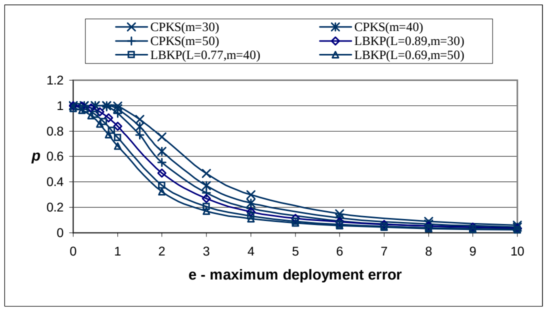
Several techniques were proposed recently to address pairwise key establishments in wireless sensor networks [5, 6], including the basic probabilistic key predistribution [6], the q-composite key predistribution [5] and the random pairwise keys scheme [5]. By taking advantage of sensors' location information, our schemes achieve better performance than these alternatives. Indeed, our closest pairwise keys scheme can be considered an extension to the random pairwise keys scheme.

Our second scheme is constructed on the basic of the polynomial based key predistribution protocol presented in [2]. The original protocol in [2] was intended to distribute group keys, and is generally not feasible in sensor networks. The scheme in [10] only uses the special case of two parties. By combining it with sensors' expected locations, our scheme achieves better performance than its counter part in [10]. A framework of key predistribution techniques based on this two-party case is proposed in [10]. Our second scheme can also be considered an instantiation of this framework with location information.

There are many other related works in sensor network security, which are mostly on key management, authentication, and vulnerability analysis. Carman, Kruus, and Matt studied the performance of a number of key management approaches in sensor networks on different hardware platform [4]. Stajano and Anderson proposed to bootstrap trust between devices through location limited channels such as physical contact [17]. Wong and Chan proposed to



(a) Fraction of compromised direct keys between non-compromised sensors v.s. number of compromised sensors. ($p = 0.33$ and $m = 40$)



(b) Probability of establishing pairwise key directly between two neighbor nodes given different e and m. The length of cell side in LBKP is configured so that it is perfectly resistant to the node captures.

Figure 7: Assume each node has available storage equivalent to 200 cryptographic keys. LBKP denotes the location-based key predistribution scheme using bivariate polynomials.

reduce the computational overhead for key exchange in low power devices with the help of a more powerful server [18]. 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 tamper-resistant hardware to protect the keys, which is not always available.

Perrig et al. developed a security architecture for sensor networks, which includes SNEP, a security primitive building block, and a broadcast authentication technique μTESLA [16], an adaption of TESLA [13–15] in sensor networks. Liu and Ning extended this technique to a multi-level key chain method to prolong the time period covered by a μTESLA instance [9]. These techniques address the critical broadcast authentication problem in sensor networks, and are also complementary to the techniques in this paper.

Wood and Stankovic identified a number of DOS attacks in sensor networks [19]. 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 [8]. Our proposed techniques may help address some of those attacks by establishing a pairwise key between two sensors.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we presented two schemes to take advantage of sensors' location information, aiming at improving pairwise key establishment in sensor networks. When sensors in a network can be deployed to the expected locations with a certain precision, our schemes provide better security and performance over the previous solutions. Our first scheme, the closest pairwise keys scheme, is resistant to node capture attacks and has no limit on the total number of sensors. Its extended version further reduces the storage overhead and simplifies the dynamic deployment of new sensors. Our second scheme, the location-based key predistribution using bivariate polynomials, employs a threshold technique and provides a trade-off between the security against node capture and the performance of establishing pairwise keys.

Several directions are worth future research. First, we would like to investigate the impact of different partitioning strategies (of the target field), such as hexagon shapes, on the performance and security of our scheme. Second, we would like to study different approaches to assigning polynomial shares and seek the optimal solution if possible.

Acknowledgment

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

7. REFERENCES

[1] S. Basagni, K. Herrin, D. Bruschi, and E. Rosti. Secure pebblenets. In *Proceedings of ACM International Symposium on Mobile ad hoc networking and computing*, pages 156–163, 2001.

[2] C. Blundo, A. De Santis, Amir 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] S. Buchegger and J. L. Boudec. Performance analysis of the CONFIDANT protocol (cooperation of nodes: Fairness in dynamic ad-hoc networks). In *Proceedings of The Third ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 226–236, June 2002.

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

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

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

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

[8] C. Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *First IEEE International Workshop on Sensor Network Protocols and Applications*, May 2003.

[9] D. Liu and P. Ning. 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*, pages 263–276, February 2003.

[10] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *10th ACM Conference on Computer and Communications Security*, October 2003.

[11] C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, and T. He. Rap: A real-time communication architecture for large-scale wireless sensor networks. In *The 8th IEEE Real-Time and Embedded Technology and Applications Symposium*, San Jose, California, September 2002.

[12] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the Sixth annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 255–265, 2000.

[13] 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.

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

[15] A. Perrig, R. Canetti, D. Song, and D. Tygar. The tesla broadcast authentication protocol. In *RSA Cryptobytes*, 2002.

[16] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J.D. Tygar. Spins: Security protocols for sensor networks. In *Proceedings of Seventh Annual International Conference on Mobile Computing and Networks*, July 2001.

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

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

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

[20] Y. Zhang and W. Lee. Intrusion detection in wireless ad hoc networks. In *Proceedings of the 6th International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 275–283, August 2000.