# A Two-Layer Key Establishment Scheme for Wireless Sensor Networks

Yun Zhou, *Student Member*, *IEEE*, and Yuguang Fang, *Senior Member*, *IEEE*

**Abstract**—In a large scale sensor network, it is infeasible to assign a unique Transport Layer Key (TLK) for each pair of nodes to provide the end-to-end security due to the huge memory cost per node. Thus, conventional key establishment schemes follow a key predistribution approach to establish a Link Layer Key (LLK) infrastructure between neighboring nodes and rely on multihop paths to provide the end-to-end security. Their drawbacks include vulnerability to the node compromise attack, large memory cost, and energy inefficiency in the key establishment between neighboring nodes. In this paper, we propose a novel key establishment scheme, called LAKE, for sensor networks. LAKE uses a $t$-degree trivariate symmetric polynomial to facilitate the establishment of both TLKs and LLKs between sensor nodes in a two-dimensional space, where each node can calculate direct TLKs and LLKs with some logically neighboring nodes and rely on those nodes to negotiate indirect TLKs and LLKs with other nodes. Any two end nodes can negotiate a TLK on demand directly or with the help of only one intermediate node, which can be determined in advance. As for the LLK establishment, LAKE is more secure under the node compromise attack with much less memory cost than conventional solutions. Due to the location-based deployment, LAKE is also energy efficient in that each node has direct LLKs with most neighbors without spending too much energy on the establishment of indirect LLKs with neighbors through multihop routing.

**Index Terms**—Sensor networks, transport layer, link layer, key establishment, node compromise.

## 1 Introduction

Security has been drawing wide interest in the area of wireless sensor networks, which usually consist of thousands of resource-limited sensor nodes deployed in a designated area without any fixed infrastructure [1], [2], [3] because of the network vulnerability to malicious attacks in unattended and hostile environments such as battlefield surveillance and homeland security monitoring [4], [5], [6], [7], [8], [9], [10]. Under such circumstances, security services such as encryption and authentication are indispensable for guaranteeing the proper operation of sensor networks.

Key management is very critical to security protocols because encryption and authentication services are based on the operations involving keys. One of the fundamental problems of key management is how to set up keys to protect connections between sensor nodes. Generally, two kinds of connections can be formed in a network. One is the one-hop connection between a pair of *neighboring* nodes. In the network stack, this one-hop connection is managed by the link layer protocol. In order to secure the link layer connection, a shared *Link Layer Key* (called LLK hereafter) needs to be established between the neighboring nodes. The other type of connection can be formed between two nodes over a multihop path. Because the two nodes are out of each other's neighborhood, this end-to-end connection is managed by the transport layer protocol instead of the link layer protocol. Therefore, a *Transport Layer Key* (called TLK hereafter) needs to be established to provide the end-to-end security.

The TLK establishment is not an easy problem. In a network of $N$ nodes, theoretically, each node can be preloaded with $N-1$ keys uniquely shared with other nodes, but the feasibility can be challenged because of the contradictory requirements between the scarce memory of sensor nodes and the large scale of sensor networks. Instead, most recent solutions [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25] relax the security requirement and target the establishment of *Link Layer Keys* (LLK) between any pair of *neighboring* nodes. In a large scale sensor network, the number of neighbors of a node is usually a small constant. Thus, it is more feasible to establish an LLK infrastructure by which to save memory resource. Based on this LLK infrastructure, two end nodes can perform secure communications over a multihop path with the help of intermediate nodes and can negotiate a TLK on demand, if needed, through the secure handshake. The LLK infrastructure can effectively prevent external attackers from accessing the network, but cannot counteract internal attackers, such as compromised nodes. Therefore, a TLK negotiated along a multilink path can be exposed if any of the intermediate nodes is compromised. Because the number of hops along a path can be large, the possibility of the TLK exposure is rather high.

Moreover, the previous LLK schemes [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25] themselves are also vulnerable to node compromise. An adversary can use the secrets in compromised nodes to

- *Y. Zhou is with the Wireless Networks Laboratory (WINET), Department of Electrical and Computer Engineering, University of Florida, 481 New Engineering Building, PO Box 116130, Gainesville, FL 32611. E-mail: yzufl@ufl.edu.*
- *Y. Fang is with the Wireless Networks Laboratory (WINET), Wireless Information Networking Group (WING), Department of Electrical and Computer Engineering, University of Florida, 435 Engineering Building, PO Box 116130, Gainesville, FL 32611. E-mail: fang@ece.ufl.edu.*

*Manuscript received 24 Jan. 2006; revised 24 July 2006; accepted 13 Nov. 2006; published online 7 Feb. 2007.*
*For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0026-0106.*
*Digital Object Identifier no. 10.1109/TMC.2007.1008.*

derive the secrets shared between other noncompromised nodes. Hence, some compromised nodes may cause many failure points in the network and destroy the entire LLK infrastructure. Another drawback of the previous LLK scheme is that they have a large memory requirement to maintain a certain level of security or connectivity.

Based on our work in [26], here, we introduce a novel scheme, called *LAKE* (*two-LAyer Key Establishment*), for the establishment of both TLKs and LLKs in sensor networks. Particularly, all sensor nodes are organized into a two-dimensional space and a trivariate polynomial is predistributed to facilitate the establishment of TLKs and LLKs in the space. To increase connectivity and reduce communication overhead, the nodes close to each other are preloaded with correlated secrets, called *shares*, derived from the trivariate polynomial. The main contributions are as follows:

1. Our LAKE effectively addresses the TLK establishment problem for sensor networks. Any two nodes can negotiate a TLK on demand directly or with the help of only one intermediate node. Though, in conventional LLK schemes, two nodes can also negotiate a TLK through a multihop path, there is more than one intermediate node that can learn the TLK. Hence, LAKE is much more secure under the node compromise attack compared with conventional proposals.
2. Compared with the conventional LLK schemes, LAKE features much less memory cost, which can be less than $1.8171\sqrt{N}$, where $N$ is the total number of nodes in the network.
3. By utilizing location information, LAKE guarantees that two neighboring nodes can establish a direct LLK with high probability. This provides energy efficiency compared with the conventional LLK schemes because the probability of indirect LLK establishment through multihop paths between two neighboring nodes is reduced.

The rest of the paper is organized as follows: Previous work is reviewed in Section 2. Details of LAKE are given in Section 3. Analysis and evaluation of LAKE compared with other schemes are carried out in Section 4. After a short discussion in Section 5, the paper is concluded in Section 6.

## 2 Related Work

Key establishment is not an easy task for WSNs. A simple global key [27] shared by all the sensor nodes is secure to external attackers that do not know the key but not internal attackers because the key can be exposed if any node is compromised. Centralized key distribution [10] may incur a large amount of communication overhead because two close nodes may have to do handshakes through a central key server at a distant place. In addition, the key server may become a potential point of failure in that the entire network is broken down if the server is corrupted by an attacker.

Due to the distributed nature of WSNs, the distributed key establishment becomes a promising technique. One method is to configure each node with a set of keys and establish a shared key between two nodes by combining the keys in the intersection of their key sets [28], [29]. The idea of the key predistribution approach is applied for WSNs in [11] and sparks many other schemes [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25]. They all target the LLK establishment, where some secrets are predistributed into sensor nodes to facilitate LLK establishment between neighboring nodes.

*Random predistribution* (RPD) schemes [11], [12], [13], [14], [15], [16], [17], [18], [19] uniformly predistribute a global set of secrets in the network so that each node has a secret subset and two neighboring nodes can achieve a *probabilistic* key agreement by the intersection of their secret subsets. If some nodes are compromised by an adversary, the secrets in those nodes are exposed to the adversary and the security of the entire network might degrade due to the uniform secret predistribution. To reduce the impact of the node compromise attack, it is wise to limit the number of secrets that each node needs to keep such that the adversary gets fewer secrets each time he/she compromises one node. Fewer predistributed secrets, however, implies a smaller probability that two neighboring nodes can establish a direct LLK, i.e., lower *local secure connectivity*, and much more communication overhead on indirect LLK establishments through multihop paths. The contradictory memory requirements by security and local secure connectivity render RPD schemes less suitable for sensor networks. Furthermore, RPD schemes do not scale well in the sense that the memory cost increases linearly with the total number of nodes in the network if a certain level of security or connectivity is desired [20].

A *deterministic* scheme, called *PIKE*, is proposed by Chan and Perrig [20]. In PIKE, all sensor nodes are put in a two-dimensional grid and one intermediate node is used to facilitate the LLK establishment between neighboring nodes. Each node is preloaded with unique pairwise keys for $2(\sqrt{N}-1)$ nodes, where $N$ is the total number of nodes, and then deployed according to a uniform distribution in the entire terrain. The low connectivity, however, indicates large energy consumption on the LLK establishment. A hypercube-based scheme is also proposed in [17]. All $N$ sensor nodes are organized into a $n$-dimensional hypercube based on their identities. Along each dimension, the scheme proposed by Blundo et al. [30] is used. Each node needs to keep shares of $n$ $t$-degree bivariate polynomials.

*Location-based predistribution* (LPD) schemes [21], [22], [23], [24], [25] use location information to localize the impact of the node compromise attack and increase connectivity by intentionally predistributing the same set of secrets in small cells. They can achieve much higher connectivity than uniform predistribution schemes because the connectivity in LPD schemes is mainly determined by node deployment information. The memory cost is proportional to the number of nodes in one cell.

Though polynomials have been used in [16], [17], [21], [30], those schemes are totally different from LAKE. Blundo et al. [30] first use a $t$-degree *bivariate* polynomial to achieve key agreement in a network. Their scheme is $t$-secure in that, in a network of $N$ nodes, the collusion of less than $t+1$ nodes cannot reveal any key held between other pairs of normal nodes. The memory cost per node to

achieve $t$-secure is $t+1$. To guarantee perfect security, a $(N-2)$-degree polynomial should be used, which means the memory cost per node is $N-1$. Therefore, it is infeasible in large scale sensor networks. The schemes [16], [17], [21] follow the probabilistic approach and use *bivariate* polynomials to achieve probabilistic LLK agreement between neighboring nodes. Particularly, they predistribute polynomial pools in the network terrain. If two neighboring nodes have polynomial(s) in common, they can calculate an LLK based on the Blundo scheme. LAKE, however, uses a *trivariate* polynomial to establish both TLKs and LLKs in a large-scale sensor network. In Section 3, we will describe LAKE in detail.

LEAP [31] assumes that attackers are not able to compromise a node during the network initialization phase and, thus, a global key is used during the network initialization phase to facilitate the key establishments between neighboring nodes. To avoid the exposure of the global key due to node compromise, the global key is deleted by each node at the end of the network initialization phase. In [32], Deng et al. proposed an OTMK scheme, which is based on LEAP but is more secure in the sense that, even if the global key is compromised, the pairwise keys between sensor nodes are still safe. The trick there is that the global key is only used to authenticate neighboring nodes but not to establish pairwise keys. Zhou et al. [33] and Zhang et al. [34] also developed node authentication and key establishment schemes based on the elliptic curve cryptography. These schemes can provide a higher security level than symmetric ones, but have more overhead. LBRS [35] divides the network into cells, each of which is associated with multiple keys. Based on its location, a node stores one key for each of its local neighboring cells and a few randomly chosen remote cells. Those keys are used to authenticate reports along the path from the source cell to the base station. These schemes are based on different assumptions and target at different purposes from the aforementioned key establishment schemes.

## 3 Two-Layer Key Establishment

In LAKE, a $t$-degree trivariate polynomial is predistributed to facilitate key establishment in a two-dimensional space. Specifically, every node is preloaded with partial information, called *share*, of a global $t$-degree trivariate polynomial. The entire network is organized into a two-dimensional space. Every node is identified in the space with its coordinates and can find other logical neighbors in the space. The predistributed shares of the global $t$-degree trivariate polynomial are used to establish TLKs and LLKs between those logical neighbors. By choosing the global polynomial degree properly, LAKE can achieve perfect resilience to the node compromise attack compared with conventional schemes.

LAKE consists of three phases: *share predistribution*, *direct key calculation*, and *indirect key negotiation*. In the share predistribution phase, shares of a global $t$-degree trivariate polynomial are predistributed among sensor nodes. All the shares cannot reveal the global polynomial even if they are captured by adversaries. Two nodes can calculate a shared key directly with the shares they hold if they are logical neighbors in the space. The indirect key negotiation phase tells how to negotiate a shared key between two nodes with the help of another node if they are not logical neighbors.

We will show that LAKE can efficiently establish both LLKs and TLKs between sensor nodes. An LLK infrastructure can be established just after network deployment. TLKs are established on demand when two end nodes need to communicate with each other.

### 3.1 System Models

#### 3.1.1 Key Agreement Model

We have proposed in [26] a scalable and deterministic key agreement model, which is based on a multivariate polynomial for large-scale distributed systems. LAKE applies a special case of [26] in wireless sensor networks, which consists of a large number of sensors. Particularly, a $t$-degree trivariate symmetric polynomial is used to achieve the key agreement between any pair of nodes. A $t$-degree trivariate polynomial is defined as

$$f(x_1,x_2,x_3)=\sum_{i_1=0}^{t}\sum_{i_2=0}^{t}\sum_{i_3=0}^{t}a_{i_1,i_2,i_3}\,x_1^{i_1}x_2^{i_2}x_3^{i_3}. \tag{1}$$

All the polynomial coefficients are chosen from a finite field $\mathbb{F}_q$, where $q$ is a prime that is large enough to accommodate a cryptographic key. Unless otherwise stated, all calculations in this paper are performed over the finite field $\mathbb{F}_q$.

A three-tuple permutation is defined as a bijective mapping

$$\sigma:\ \{1,2,3\}\longrightarrow\{1,2,3\}. \tag{2}$$

By choosing all the coefficients according to

$$a_{i_1,i_2,i_3}=a_{i_{\sigma(1)},i_{\sigma(2)},i_{\sigma(3)}} \tag{3}$$

for any permutation $\sigma$ of $\{1,2,3\}$, we can obtain a symmetric polynomial in that

$$f(x_1,x_2,x_3)=f(x_{\sigma(1)},x_{\sigma(2)},x_{\sigma(3)}). \tag{4}$$

At first, every node should have two credentials, which are *positive* and *pairwise different* integers. Suppose node $u$ has credentials $(u_1,u_2)$ and node $v$ has $(v_1,v_2)$. Before node deployment, a *polynomial share* $f(u_1,u_2,x_3)$ is assigned to $u$ and another share $f(v_1,v_2,x_3)$ to $v$. By assigning polynomial shares, we mean that the coefficients of $t$-degree univariate polynomials $f(u_1,u_2,x_3)$ and $f(v_1,v_2,x_3)$ are loaded into nodes $u$'s and $v$'s memories, respectively.

If the credentials of nodes $u$ and $v$ have one element in common, i.e., $u_1=v_1,u_2\neq v_2$ or $u_1\neq v_1,u_2=v_2$, then node $u$ and node $v$ can have a shared key. Suppose the $i$th credentials of $u$ and $v$ are different, where $i=1$ or $i=2$. Node $u$ can take $v_i$ as the input to its own share $f(u_1,u_2,x_3)$ and node $v$ can take $u_i$ as the input to its share $f(v_1,v_2,x_3)$. Then, the desired shared key between nodes $u$ and $v$ can be established as

$$K_{uv}=f(u_1,u_2,v_i)=f(v_1,v_2,u_i), \tag{5}$$

where $i=1$ or $i=2$.

In fact, nodes $u$ and $v$ achieve the key agreement by a marginal $t$-degree bivariate polynomial,

$$f_{c_j}(x_i, x_3) = f(c_j, x_i, x_3), \tag{6}$$

where $c_j$ for $j \neq i$ and $i, j \in \{1, 2\}$ is the common credential between nodes $u$ and $v$.

#### 3.1.2 Network Model

It has been shown in [21], [22], [23], [24], [25] that utilizing deployment information can achieve higher connectivity. So, even though our key agreement model can achieve deterministic key agreement between any pair of nodes (as is shown in [26]), we consider incorporating deployment information into LAKE.

The entire network is divided into $N_1$ nonoverlapping square cells and each cell includes $N_2$ sensor nodes. Each node in the network is identified by a coordinate $(n_1, n_2)$ in the two-dimensional space, where

$$n_i = 0, 1, \ldots, N_i - 1, i \in \{1, 2\},$$

and we may use the coordinate $(n_1, n_2)$ as the *node ID* and the index $n_1$ as the *cell ID*.

Cell IDs are assigned in a fixed order such that each cell ID acts like a coordinate in a two-dimensional plane. We may allocate $2h$ higher bits from the node ID field for the cell ID. The $2h$ bits are divided into a pair of integers $(i, j)$, where $i$ is the row index and $j$ is the column index of the cell. Hence, each cell ID reflects the location information of the corresponding nodes. This information is coarse, so we can only tell in which area a node with a given cell ID resides. The node deployment in each cell may follow any probabilistic distribution, such as Gaussian [15], [24], [36] or uniform [21], [23], [25]. We assume Gaussian distribution here and evaluate its influence in Section 4.

Our key agreement model is deterministic, so every node knows with which other nodes it can establish a shared key directly. If two nodes cannot calculate a shared key directly, they rely on one intermediate node to negotiate an indirect key. Just like in previous work [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], we assume that the underlying routing protocol can correctly route key negotiation messages over multihop paths between those peer nodes.

Fig. 1 illustrates an example of the network model. There are 64 cells in the network. Each cell consists of $N_2$ nodes. We assign cell IDs in an order from left to right and from top to down. Every node can be located by the cell ID in its node ID. For example, node $(0, n_2)$ is in the most up-left cell, and node $(63, n_2)$ is in the most down-right cell. Other examples of nodes are also depicted in Fig. 1.

#### 3.1.3 Adversary Model

Because of the broadcast nature of radio communications, adversaries can easily eavesdrop any messages, either nonencrypted or encrypted, transmitted over the air between nodes. Moreover, it is also unrealistic and uneconomical to employ tamper-resistant hardware to secure the cryptographic materials in each individual node because of cost constraints. Even if tamper resistant devices are available, they are still not able to guarantee perfect security of secrets [37]. Hence, adversaries may capture any node and compromise the secrets stored in the node. Furthermore, adversaries can use the compromised secrets to derive more secrets shared between other noncompromised nodes. It means that the node compromise attack is unavoidable. What we can do is to reduce the impact on other normal nodes as much as possible. In LAKE, we attempt to reduce the probability that the keys shared between noncompromised nodes are exposed when some nodes have already been compromised.

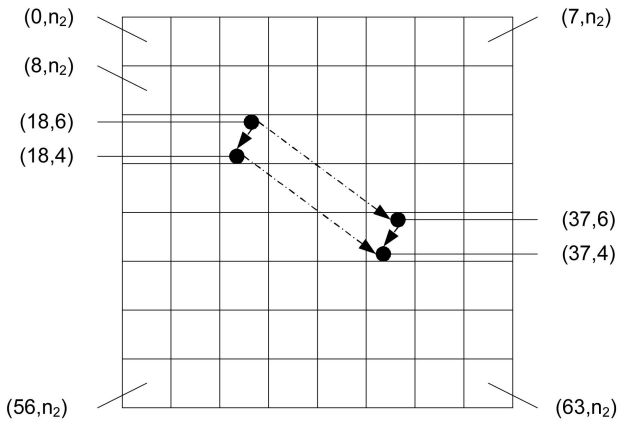



Fig. 1. A two-dimensional sensor network.

### 3.2 Share Predistribution

Before network deployment, a global $t$-degree trivariate polynomial is chosen as is stated in Section 3.1. This polynomial is used to derive shares for sensor nodes.

To establish keys, every node should have two credentials $(c_1, c_2)$, which are positive and pairwise different. These credentials can be created and preloaded into nodes before deployment. However, it requires additional memory space per node. Fortunately, the two credentials can be derived from node ID by a bijection, i.e.,

$$\begin{cases} c_1 = n_1 + 1 + N_2 \\ c_2 = n_2 + 1, \end{cases} \tag{7}$$

where $n_i = 0, 1, \ldots, N_i - 1$ for $i = 1, 2$. Thus, the two credentials are drawn from different zones $[N_2 + 1, N_1 + N_2]$ and $[1, N_2]$, respectively, which guarantee they are positive and pairwise different. Besides, by doing this mapping, each node needs to store only $N_2$ instead of two credentials.

Every node in the network is assigned a polynomial share

$$\begin{aligned} f(c_1, c_2, x_3) &= f(n_1 + 1 + N_2, n_2 + 1, x_3) \\ &= \sum_{i_1=0}^{t} \sum_{i_2=0}^{t} \sum_{i_3=0}^{t} a_{i_1,i_2,i_3} (n_1 + 1 + N_2)^{i_1} (n_2 + 1)^{i_2} x_3^{i_3}. \end{aligned} \tag{8}$$

Hence, every node in the network needs to keep only a $t$-degree univariate polynomial that has $t + 1$ coefficients over a finite field $\mathbb{F}_q$. Those coefficients are preloaded into

every node's memory before deployment and used to establish keys after deployment.

### 3.3 Direct Key Calculation

As is stated in Section 3.1, two nodes can calculate a shared key directly if they have a credential in common, i.e., a common index in their node IDs. We will call one of the two nodes a *level-$i$ neighbor* of the other if their $i$th indices in their IDs are different and the other indices are the same. Obviously, every node can establish shared keys with its neighbors at level 1 (intercell) and level 2 (intracell).

In the two-dimensional network, all nodes in each cell are level-2 neighbors because they have the same cell ID, and each node has a level-1 neighbor in each of other cells. For example, in the two-dimensional network in Fig. 1, node (18,4) and node (18,6) are level-2 neighbors and they can calculate a shared key directly. Node (18,4) and node (37,4) are level-1 neighbors and they can also calculate a shared key directly.

All nodes can calculate direct keys on their own without interaction with other logical neighbors. Each direct key between two logical neighboring nodes is only secret to them. An adversary cannot learn the direct key unless he/she knows the corresponding polynomial share.

### 3.4 Indirect Key Negotiation

If two nodes have no common indices in their IDs, they cannot calculate a shared key directly because they are not logical neighbors. This happens when the two nodes reside in different cells and they have different indices in their cells, respectively. In this case, one node can find in its cell a level-2 neighbor, which is also a level-1 neighbor of the other node. Then, the intermediate node can act as an *agent* to facilitate a shared key negotiation between the two end nodes.

There are two agent nodes that can help the indirect key negotiation. Suppose node $u$ with ID $(u_1, u_2)$ and node $v$ with ID $(v_1, v_2)$, where $u_1 \neq v_1$ and $u_2 \neq v_2$, need to negotiate a shared key. Either node $(u_1, v_2)$ or node $(v_1, u_2)$ can act as an agent because either one is the common neighbor of nodes $u$ and $v$. Then, an indirect key can be established through the following protocol:

$$\begin{aligned} u \rightarrow a: \quad & \langle a, u, n_u, \{\langle v, u, K_{uv}\rangle\}_{K_{ua}}, H(a \parallel u \\ & \parallel n_u \parallel \{\langle v, u, K_{uv}\rangle\}_{K_{ua}} \parallel K_{ua})\rangle, \\ a \rightarrow v: \quad & \langle v, a, n_a, \{\langle v, u, K_{uv}\rangle\}_{K_{av}}, H(v \parallel a \\ & \parallel n_a \parallel \{\langle v, u, K_{uv}\rangle\}_{K_{av}} \parallel K_{av})\rangle, \end{aligned}$$

where $a$ is an agent node, $n_u$ and $n_a$ are nonces used to counteract replay attacks, $K_{uv}$ is the indirect key between node $u$ and node $v$, $K_{ua}$ and $K_{av}$ are direct keys shared with the agent $a$, "$\{\cdot\}_{\{\cdot\}}$" is the encryption operation, $H(\cdot)$ is a hash function that generates a *Message Authentication Code* for authentication and integrity checking, and "$\parallel$" is the concatenation operator. After verifying the authenticity and the integrity of the key $K_{uv}$, the agent node $a$ forwards the key to node $v$ and immediately deletes it so that it cannot be revealed later.

For example, in Fig. 1, there are two secure paths between node (18,6) and node (37,4) and a shared key can be negotiated through either of the secure paths with the help of the agent nodes (18,4) or (37,6).

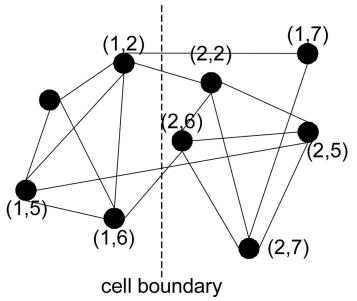



Fig. 2. LLK establishment.

### 3.5 LLK Establishment

Given node density and radio radius in a large-scale sensor network, the number of neighbors of a node is usually small. Each node may establish LLKs for all neighbors and keep those LLKs in its memory for future use. This can be done just after node deployment because each node has been preloaded with a polynomial share which can help key establishment.

When two neighboring nodes are from the same cell, i.e., have the same cell index, they can apply the direct key calculation to establish an LLK. Due to the deployment knowledge, we can expect that each node can establish LLKs directly with most of its neighboring nodes because they are almost from the same cell.

If two neighboring nodes are from different cells but they are level-1 neighbors, then they can calculate a direct LLK, just like nodes (1,2) and (2,2) in Fig. 2. Even if two level-1 neighbors are far away from each other, like nodes (1,5) and (2,5), they can always calculate a shared key independently.

The keys between level-1 neighbors can act as bridges between two cells. A node in one cell can go through any of the bridges to negotiate keys with nodes in the other cell. In Fig. 2, for example, node (1,2) can negotiate an indirect LLK with node (2,6) through either node (2,2) or node (1,6).

Due to the deployment error, some nodes may reside outside of their supposed cells. In Fig. 2, for example, node (1,7) needs to establish LLKs with neighboring nodes (2,2), (2,5), and (2,6). In this case, node (1,7) can carry out indirect key negotiation through its level-1 neighbor (2,7). Of course, node (2,7) may be a multihop away from node (1,7), but the underlying routing protocol can route key negotiation messages between them, just as is shown in previous work [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25].

Communication overhead is a concern in the indirect LLK negotiation. LAKE includes deployment information into the establishment of LLKs, thus, each node may calculate direct LLKs with almost all of its neighbors. This high local secure connectivity is desirable because it means that each node does not need to spend too much energy on the establishment of indirect LLKs with neighbors through multihop routing. Conventional LLK schemes with uniform key predistribution have more energy consumption in

terms of lower local secure connectivity. In Section 4, we will evaluate the secure connectivity assuming Gaussian distribution for node location in each cell.

### 3.6 TLK Establishment

Due to the huge number of nodes in the network, it is impossible to establish a TLK for each pair of nodes and store the TLK in the pair of nodes during the network initialization phase. A dynamic establishment of TLKs is very promising in large-scale sensor networks. Generally, a TLK should be dynamically established on demand during the handshake procedure between any pair of nodes when they want to communicate with each other.

Similar to the LLK establishment, each node can establish a direct TLK for each of the other nodes in its cell because they are level-2 neighbors. As each node has a level-1 neighbor in each of other cells in the network, it can establish a direct TLK for the level-1 neighbor in that cell (like nodes (18,6) and (37,6) in Fig. 1). Then, it can rely on the level-1 neighbor as an agent to establish indirect TLKs with other nodes in that cell (in Fig. 1) and node (18,6) can negotiate an indirect TLK with node (37,4) through node (37,6)). Due to the deployment error, there is a possibility that node (37,6) is not in cell 37, but the underlying routing protocol can relay key negotiation messages between these nodes as is assumed in previous work [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25].

If a *secure link* is defined as the communication path between two nodes that have a shared key, where the secure link may be one-hop or multihop, LAKE can achieve the TLK agreement through a *secure path* consisting of no more than two secure links, which means that at most one agent is needed to facilitate the TLK establishment between any two end nodes. Each secure path in most conventional schemes, which target LLK establishment, usually consists of more than two secure links, and the length of the secure path is difficult to determine because it depends on not only the underlying routing protocol, but also the establishment of direct keys between neighboring nodes, especially in large-scale networks. Thus, most conventional schemes are more vulnerable to the node compromise attack than LAKE.

## 4 Security Analysis and Performance Evaluation

In this section, we carry out some performance evaluation on the memory cost, the resilience to the node compromise attack, the local secure connectivity, and the computation overhead.

### 4.1 Metrics

#### 4.1.1 Memory Cost

We will calculate how many memory units per node are necessary for its polynomial share, where each memory unit can accommodate a cryptographic key in conventional schemes or a polynomial coefficient. Due to the resource constraint of sensor nodes, the small memory cost is desirable.

#### 4.1.2 Resilience to the Node Compromise Attack

Usually, it is unavoidable to prevent an adversary from launching the node compromise attack. We can do nothing to rescue those compromised nodes. However, a good scheme should reduce the impact of the node compromise attack on other normal nodes as much as possible. By compromising a node, the adversary can learn the keys that the compromised node uses to communicate with other nodes, but he/she should not learn other keys that the compromised node does not know so that the communications between other normal nodes are still safe. The additional key exposure probability is used here to evaluate the resilience to the node compromise attack. Specifically, the probability that the keys shared between other noncompromised nodes are exposed should be as small as possible when the adversary has already compromised some nodes.

#### 4.1.3 Local Secure Connectivity

Local secure connectivity is the probability that two neighboring nodes establish a direct key, i.e., the portion of neighbors with which a node can establish direct keys. It is directly related to the communication overhead of key establishments. In sensor networks, high local secure connectivity is desirable because it means that each node does not need to spend too much energy on the establishment of indirect keys with neighbors through multihop routing, thus saving a lot of communication overhead.

#### 4.1.4 Computation Overhead

We will evaluate the computation overhead for each node to calculate a direct key. LAKE uses efficient symmetric key techniques to achieve key agreements and is thus viable on resource constrained sensor platforms.

### 4.2 Memory Cost

All nodes in the network hold partial information of a global $t$-degree trivariate polynomial to achieve key agreement. If an adversary compromises some nodes and gathers many pieces of those partial information, he/she can use those pieces of information to derive keys shared between other normal nodes. The adversary can launch two scales of the node compromise attack: *level scale* and *network scale*.

Each node has $N_2 - 1$ level-2 neighbors in its cell and $N_1 - 1$ level-1 neighbors from other cells, and it can establish direct keys with these logical neighbors through $t$-degree bivariate polynomials, which are the marginal polynomials of the global polynomial (refer to Section 3.1). If the adversary compromises enough nodes at one level, the bivariate polynomial of the level may be exposed in the level scale attack so that the adversary can calculate the direct keys between any pairs of nodes in the level.

Furthermore, the adversary can launch the network scale attack to compromise nodes at more than one level. If the adversary gets enough information, he/she may even expose the global $t$-degree trivariate polynomial; thus, the secure infrastructure of the entire network is destroyed and the adversary may calculate the key between any pair of nodes in the network. The hardness of exposing the global polynomial depends on the polynomial degree $t$. Thus, the

value of $t$ should be large enough to provide the secrecy of the global polynomial so that no matter how many nodes the adversary compromises, the global polynomial is still secret to the adversary. The value of $t$ is directly related to the memory cost because each node keeps a $t$-degree univariate polynomial which has $t+1$ coefficients. Hence, the minimal $t$ which can provide the secrecy of the global polynomial should be chosen.

#### 4.2.1 Level Scale

Let us consider the level scale node compromise attack. Any pair of nodes at one level can achieve key agreement by a marginal $t$-degree bivariate polynomial,

$$f_{c_j}(x_i, x_3) = f(c_j, x_i, x_3), \tag{9}$$

where $c_j$ for $j \neq i$ is the common credential between the pair of nodes (refer to Section 3.1). It has been shown in [30] that a $t$-degree bivariate polynomial is $t$-*secure* in that the coalition between less than $(t+1)$ nodes holding shares of the $t$-degree bivariate polynomial cannot reconstruct the polynomial. To guarantee that any pair of nodes among a level of $N_i$ nodes has a direct key that is unsolvable by other $N_i - 2$ nodes, an $(N_i - 2)$-secure bivariate polynomial should be used. Hence, the degree of polynomial should satisfy

$$0 \leq N_i - 2 \leq t, \qquad i = 1, 2. \tag{10}$$

#### 4.2.2 Network Scale

Let us consider the attack at the network scale. If the adversary compromises all $N_i$ nodes at one level, he/she can construct $\frac{N_i(N_i+1)}{2}$ equations, i.e.,

$$\begin{cases} f_2(c_{i,0},\ c_{i,0}) = K_{0,0} \\ \vdots \\ f_2(c_{i,0},\ c_{i,N_i-1}) = K_{0,N_i-1} \\ f_2(c_{i,1},\ c_{i,1}) = K_{1,1} \\ \vdots \\ f_2(c_{i,N_i-1},\ c_{i,N_i-1}) = K_{N_i-1,N_i-1}, \end{cases} \tag{11}$$

where $c_{i,j}$ for $j = 0, 1, \ldots, N_i - 1$ is the credential of the $j$th node at level $i$ for $i = 1, 2$, $K_{j_1,j_2}, j_1 \neq j_2$ is the direct key between the $j_1$th and the $j_2$th nodes at level $i$ and $K_{j,j}$ is calculated with the polynomial share of the $j$th node. The total number of nodes is

$$N = N_1 \cdot N_2. \tag{12}$$

Suppose that all of the $N$ nodes are compromised. The total number of equations the adversary can construct is

$$\begin{aligned} N_e &= N_2 \cdot \frac{N_1(N_1+1)}{2} + N_1 \cdot \frac{N_2(N_2+1)}{2} \\ &= \frac{N_1 N_2}{2}(N_1 + N_2 + 2). \end{aligned} \tag{13}$$

The number of distinct coefficients of a $t$-degree trivariate symmetric polynomial is [30]

$$N_c = \binom{t+3}{3}. \tag{14}$$

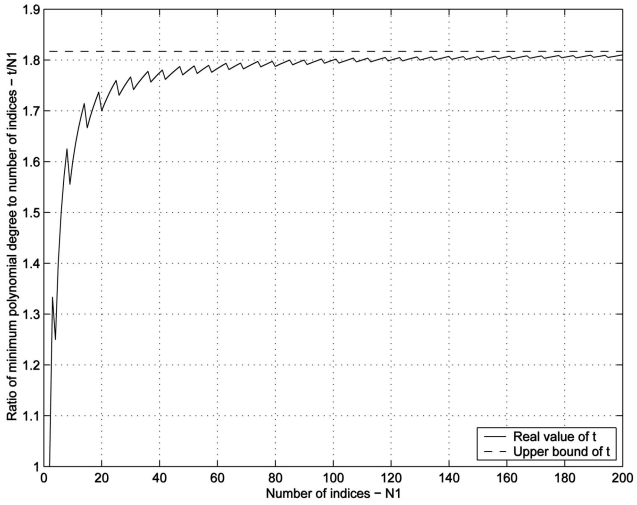



Fig. 3. Minimum required polynomial degree versus number of nodes at each level.

Thus, to guarantee the secrecy of the global polynomial, the following condition should be satisfied, i.e.,

$$N_e \leq N_c \Longrightarrow \frac{N_1 N_2}{2}(N_1 + N_2 + 2) \leq \binom{t+3}{3}. \tag{15}$$

#### 4.2.3 Choice of $t$

To sum up, the degree $t$ of the global polynomial should satisfy the following conditions, i.e.,

$$\begin{cases} 0 \leq N_i - 2 \leq t, \quad i = 1, 2 \\ \frac{N_1 N_2}{2}(N_1 + N_2 + 2) \leq \binom{t+3}{3}. \end{cases} \tag{16}$$

Hence, to minimize the memory cost, we should use the polynomial which has the minimum degree satisfying the above inequalities.

A common case to design a network is to let $N_1 = N_2$. Thus,

$$\begin{aligned} {N_1}^2(N_1+1) &\leq \left(\frac{t}{3}+1\right) \cdot \left(\frac{t}{2}+1\right) \cdot (t+1) \\ \Longrightarrow {N_1}^3 + {N_1}^2 &\leq \frac{t^3}{3!} + t^2 + \frac{11t}{6} + 1. \end{aligned} \tag{17}$$

The minimum polynomial degree $t^*$ should satisfy this inequality. In another way, when

$$t \geq N_1 \sqrt[3]{3!}, \tag{18}$$

(17) is automatically satisfied. Because $\binom{t+3}{3}$ is a monotonic increasing function of $t$, the solution of (17) should be $[t^*, \infty)$. Because the solution of (18) is also the solution of (17), the minimum polynomial degree $t^*$ can be bounded as

$$t^* \leq \sqrt[3]{3!}\, N_1 = 1.8171\, N_1. \tag{19}$$

Fig. 3 gives the ratio of the minimum $t^*$ to the number of nodes at level-1 and its upper bound.

Because each node keeps $t+1$ coefficients of a $t$-degree univariate polynomial, the memory cost per node is less than $1.8171\sqrt{N} + 1$, where $N$ is the total number of nodes in the network.

TABLE 1
Memory Cost of Different Schemes

| Schemes | Memory Cost | Memory Cost for Secrecy of Direct Keys |
|---|---|---|
| key-pool-based [11]–[14], [17]–[19], [22] | $m$ | |
| space-pool-based [15], [16], [23] | $\mathcal{O}(\lambda(t+1))$ | $\mathcal{O}(\lambda(N-1))$ |
| PIKE-2D [20] | $2(\sqrt{N}-1)$ | $2(\sqrt{N}-1)$ |
| Hypercube-2D [17] | $2(t+1)$ | $2(\sqrt{N}-1)$ |
| LBKP [21] | $5(t+1)$ | $5(5\sqrt{N}-1)$ |
| Zhou *et al.* [24] | $6(t+1)$ | $6(2\sqrt{N}-1)$ |
| Zhou *et al.* [25] | $3(t+1)$ | $3(2\sqrt{N}-1)$ |
| LAKE | $t+1$ | $< 1.8171\sqrt{N}+1$ |

*The second column gives the normal memory cost. The third column gives how many memory units are necessary to provide secrecy for direct keys.*

We compare the memory cost per node of our LAKE with some typical schemes in Table 1. The second column in Table 1 gives the normal memory cost of each scheme. In key-pool-based schemes [11], [12], [13], [14], [17], [18], [19], [22], each node keeps $m$ keys out of a global or local key pool. Du et al.'s [15], Liu and Ning's [16], and Huang et al.'s [23] schemes replace key pools with space pools of matrix or polynomials of degree $t$. In PIKE [20], each node is preloaded with unique pairwise keys for $2(\sqrt{N}-1)$ nodes, where $N$ is the total number of nodes in the network. The hypercube scheme [17] uses a higher dimensional grid. Unlike PIKE, hypercube uses a $t$-degree bivariate polynomial for each dimension. For fair comparison, we assume a two-dimensional grid for hypercube. Therefore, the memory cost of hypercube is $2(t+1)$. In LBKP [21], each node is preloaded with five polynomial shares, each of which has a degree of $t$. Zhou et al.'s schemes [24], [25] have a memory cost of $6(t+1)$ and $3(t+1)$, respectively. In LAKE, unlike in previous work, each node needs to keep only a $t$-degree univariate polynomial and, thus, the memory cost is only $t+1$.

The third column in Table 1 gives how many memory units are necessary to provide secrecy for direct keys, i.e., no matter how many nodes are compromised, the direct keys among noncompromised nodes are still safe. Key-pool-based schemes [11], [12], [13], [14], [17], [18], [19], [22] cannot provide secrecy because, each time an adversary compromises one more node, he/she knows more keys in the global or local key pools. In Du et al.'s [15], Liu and Ning's [16], and Huang et al.'s [23] schemes, the degree of each matrix or polynomial must be set as $t = N-2$ to avoid the exposure of direct keys. So, their memory cost is on the order of $N$. In PIKE [20], all those $2(\sqrt{N}-1)$ keys are preloaded and unique, so any of the keys is secure even if other keys are compromised. In a two-dimensional hypercube [17], each dimension has $\sqrt{N}$ nodes. In order to protect direct keys, the polynomial degree must be set as $t = \sqrt{N}-2$ and, thus, the memory cost is $2(\sqrt{N}-1)$. Suppose that LBKP [21], Zhou et al.'s schemes [24], [25], and LAKE use the same network configuration, where the entire network is divided into $\sqrt{N}$ cells and each cell consists of $\sqrt{N}$ nodes. In LBKP [21], to guarantee that each bivariate polynomial is secret, the degree should be no less than $5\sqrt{N}-2$ because each bivariate polynomial is used in its home cell and four adjoining cells. Similarly, the memory costs of Zhou et al.'s schemes [24], [25] are $6(2\sqrt{N}-1)$ and $3(2\sqrt{N}-1)$, respectively. However, the memory cost of LAKE is less than $1.8171\sqrt{N}+1$.

### 4.3 Resilience to the Node Compromise Attack

By launching the node compromise attack, an adversary may easily obtain all secrets stored in the compromised nodes. Usually, it is impossible to prevent this kind of attack due to the lack of tamper-proof hardware. Furthermore, the adversary may use the compromised secrets to derive the direct keys belonging to other pairs of normal nodes. In addition, by compromising some nodes, the adversary can also obtain the messages passing through these nodes. This may also lead to the exposure of indirect keys. Here, we can use the *additional key exposure probability* to evaluate the resilience to the node compromise attack.

#### 4.3.1 Additional Direct Key Exposure Probability

By choosing the global polynomial degree $t$, we can achieve the secrecy of the global polynomial, i.e., no matter how many nodes an adversary compromises, he/she cannot calculate the direct keys belonging to other pairs of noncompromised nodes. Hence, the additional direct key exposure probability of LAKE is 0.

In conventional key-pool-based or space-pool-based schemes [11], [12], [13], [14], [15], [16], [17], [18], [19], [22], [23], every time an adversary compromises one more nodes, he/she obtains more information about the global key pool or space pool, which means that more keys are compromised. For example, in E-G [11], the additional direct key exposure probability can be calculated as [12], [22]

$$P_c = 1 - \left(1 - \frac{M}{S}\right)^x, \tag{20}$$

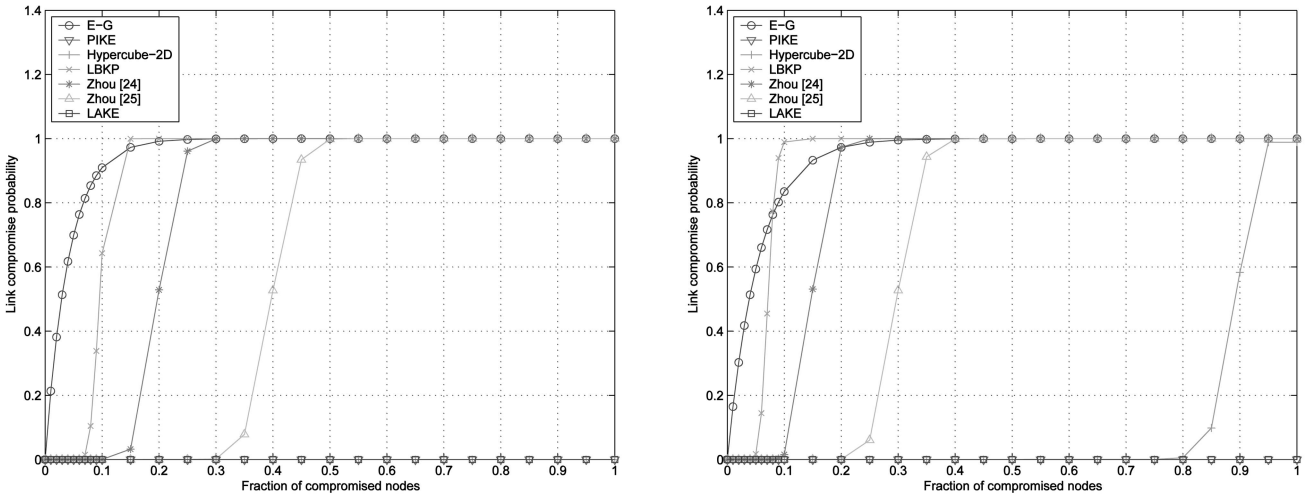



Fig. 4. Probability of direct key exposure versus fraction of compromised nodes. The x-axis is the fraction of compromised nodes in the network. The y-axis is the probability of direct key exposure, i.e., the probability of link compromise. (a) $M = 240$. (b) $M = 180$.

where each node randomly selects a key subset of size $M$ from a global key set of size $S$ and the number of compromised nodes is $x$. We can see that it is an increasing function of $x$.

PIKE [20] can also achieve the zero additional direct key exposure probability because of the predistribution of unique direct keys.

Hypercube-2D [17], LBKP [21], and Zhou et al. [24], [25] use a $t$-degree bivariate polynomial to achieve key agreement. Suppose that there are $A$ nodes sharing a $t$-degree bivariate polynomial and $N$ is the total number of nodes in the network. Given $x$ compromised nodes, the probability that $i$ out of $A$ nodes were compromised is

$$P_c(i) = \frac{\binom{A}{i}\binom{N-A}{x-i}}{\binom{N}{x}} \tag{21}$$

and, thus, the probability that the polynomial was compromised is given by

$$P_c = \sum_{i=t+1}^{A} P_c(i). \tag{22}$$

Supposing that each node has a memory size of $M$ units for cryptographic materials, each memory unit can accommodate a cryptographic key or a polynomial coefficient, and each node must keep $B$ polynomial shares, the probability of exposing a polynomial can be rewritten as

$$P_c = \sum_{i=\lfloor \frac{M}{B} \rfloor}^{A} P_c(i). \tag{23}$$

Here, the values of $A$ and $B$ are $\sqrt{N}$ and 2 for Hypercube-2D [17], $5N_c$ and 5 for LBKP [21], $2N_c$ and 6 for Zhou et al. [24], and $2N_c$ and 3 for Zhou et al. [25], where $N_c$ is the number of nodes in one cell.

**Example.** Suppose 10,000 nodes are deployed in an area $2,000 \times 2,000\ \mathrm{m}^2$. The global key pool of E-G [11] is set to 100,000. PIKE [20] and Hypercube [17] use a two-dimensional grid. For LBKP [21], Zhou et al. [24], [25], and LAKE, there are 100 cells and, thus, the number of nodes per cell is 100. Suppose that each node has a memory size of $M$ units for cryptographic materials and each memory unit can accommodate a cryptographic key or a polynomial coefficient. Fig. 4 gives the additional direct key exposure probability according to the fraction of compromised nodes when $M = 240$,180. We observe that LAKE outperforms other schemes with the zero probability of the additional direct key exposure. When there is more memory resource ($M = 240$), Hypercube-2D [17] can also give the zero probability of the additional direct key exposure. However, when memory resource is limited ($M = 180$), Hypercube-2D [17] becomes vulnerable to node compromise.

#### 4.3.2 Additional Indirect Key Exposure Probability

Every node needs an agent node to establish indirect LLKs and TLKs with other nodes. If the agent node is compromised, the indirect keys are exposed. Suppose $x$ out of $N$ nodes in the network are compromised. The probability of the indirect key exposure is

$$P_c = 1 - \frac{\binom{N-1}{x}}{\binom{N}{x}} = \frac{x}{N}. \tag{24}$$

PIKE-2D [20] and Hypercube-2D [17] can achieve the same probability of the indirect key exposure because it also relies one agent node to establish pairwise keys between neighboring nodes. In other schemes [11], [12], [13], [14], [15], [16], [17], [18], [19], [22], [23], [21], [24], [25], two nodes have to rely on a secure path consisting of multiple agent nodes to establish an indirect key. It is very difficult to determine those agent nodes because it depends on not only the underlying routing protocol, but also the establishment of direct keys between neighboring nodes, especially in large-scale networks. For example, in E-G [11], the secure path between two neighboring nodes consists of two or

three agent nodes and the secure path between any two end nodes consists of more than 11 agent nodes on average [11]. Suppose the secure path between two nodes in E-G and LBKP consists of $h$ agent nodes. The probability that the indirect key between the two end nodes is exposed can be calculated as

$$p_c = 1 - \frac{\binom{N-h}{x}}{\binom{N}{x}} \approx 1 - \left(1 - \frac{x}{N}\right)^h \approx \frac{xh}{N}, \quad (25)$$

where $N \gg h > 1$. Thus, LAKE is more resilient to the node compromise attack.

### 4.4 Local Secure Connectivity

Every node can calculate direct LLKs with some neighbors and establish indirect LLKs with other neighbors through one agent node. The local secure connectivity is directly related to the communication overhead of key establishment. If a node has high probability to calculate direct LLKs, it can save a lot of communication overhead on the establishment of indirect LLKs through multihop routing. Hence, high local secure connectivity, which is the probability of establishment of direct LLKs, is desirable in sensor networks.

In the schemes [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], key materials are uniformly predistributed in the network. It is highly possible that two nodes with correlated key materials cannot establish a direct LLK because they are far away from each other. For example, in the E-G scheme [11], each node randomly selects $M$ keys from $S$ keys; thus, the local secure connectivity of E-G is

$$1 - \binom{S-M}{M} \Big/ \binom{S}{M} \approx 1 - \left(1 - \frac{M}{S}\right)^M \approx \frac{M^2}{S},$$

where $S \gg M$. In PIKE-2D [20] and Hypercube-2D [17], each node keeps pairwise keys with $2(\sqrt{N}-1)$ nodes; thus, the local secure connectivity of these two schemes is $2(\sqrt{N}-1)/N \approx \frac{2}{\sqrt{N}}$. The low connectivity will incur significant communication overhead.

It has been shown that deployment knowledge can be used to increase the connectivity [21], [22], [23], [24], [25]. By intentionally predistributing the same set of secrets in small cells, they can achieve much higher connectivity than uniform predistribution schemes. Though our key agreement model still works in uniform predistribution scenarios, we consider deployment knowledge here to further increase the local secure connectivity.

Due to deployment errors, we cannot expect each node to reside at the predetermined location. Rather, the node deployment in each cell follows some probabilistic distribution. In order to evaluate the influence of deployment knowledge, we use the Gaussian distribution [15], [24], [36] in our simulation. Particularly, the location of each node follows the distribution

$$p(x,y) = \frac{1}{2\pi\sigma^2} \exp \frac{-((x-x_c)^2 + (y-y_c)^2)}{2\sigma^2}, \quad (26)$$

where $(x_c, y_c)$ is the center of the cell in which the node resides and $(x, y)$ is the real location of the node.

TABLE 2
Local Secure Connectivity of Different Schemes

| Schemes | Local Connectivity |
|---|---|
| E-G [11] | 0.2797 |
| PIKE-2D [20], Hypercube-2D [17] | 0.0276 |
| LBKP [21] | 0.9999 |
| Zhou *et al.* [24] | 0.9960 |
| Zhou *et al.* [25] | 0.9985 |
| LAKE | 0.5317 |

We use the same configuration parameters in Section 4.3 in our simulation. There are 10,000 nodes deployed in an area of $2,000 \times 2,000\ \mathrm{m}^2$. All the schemes evaluated here can store $M = 200$ keys. The node radio radius is $150\ \mathrm{m}$, which corresponds to MICA2 capability [38]. The global key pool of E-G [11] is set 100,000. The schemes E-G [11], PIKE-2D [20], and Hypercube [17] use the uniform predistribution. As for the location-based schemes LBKP [21], Zhou et al. [24], [25], and LAKE, there are 100 cells and, thus, the number of nodes per cell is 100. The intercell distance (the distance between the centers of neighboring cells) is set to $200\ \mathrm{m}$. The standard derivation is set to $\sigma = 50\ \mathrm{m}$. Under these configurations, we simulate a sensor network, find the local secure connectivity of each node, and average it over all the nodes. The average local secure connectivity is given in Table 2.

We observe that the local secure connectivity for the uniform predistribution schemes [11], [20], [17] is very low. The location-based schemes [21], [24], [25] have much higher connectivity because all the nodes in neighboring cells are predistributed with correlated key materials. LAKE has lower local secure connectivity than the location-based schemes [21], [24], [25] because, in LAKE, only the nodes in one cell have correlated key materials and each node can establish a direct key with only one node in another cell. However, LAKE still has much higher local secure connectivity than the uniform predistribution schemes such as E-G [11], PIKE-2D [20], and Hypercube [17].

### 4.5 Computation Overhead

LAKE is based on the symmetric key technology, where a global $t$-degree trivariate symmetric polynomial is used to build up a secure infrastructure. Each sensor node can calculate a key by using a $t$-degree univariate polynomial, which is a share of the global polynomial. It has been shown in Hypercube [17] that the polynomial evaluation is comparable with conventional symmetric key primitives such as Message Authentication Code based on RC5 or SkipJack. To calculate a key, each node should calculate $2t-1$ modular multiplications over $\mathbb{Z}_q^*$: $t-1$ for $x^2, \ldots, x^t$ and $t$ for $b_1 x, b_2 x^2, \ldots, b_t x^t$. Under the symmetric key technology, the length of $q$ is usually 64 bits or 128 bits. Suppose the same configuration parameters in Section 4.3

TABLE 3
Computation Overhead of Different Schemes

| Schemes | #. of polynomial shares | polynomial degree | #. of modular multiplications |
|---|---|---|---|
| Hypercube-2D [17] | 2 | 98 | 390 |
| LBKP [21] | 5 | 498 | 3875 |
| Zhou *et al.* [24] | 6 | 198 | 2370 |
| Zhou *et al.* [25] | 3 | 198 | 1185 |
| LAKE | 1 | $< 181$ | $< 361$ |

*The second column gives the number of polynomial shares each node needs to keep. The third column gives the maximum polynomial degree. The fourth column gives the number of modular multiplications each node needs to perform to calculate a direct key.*

are used here, where total number of nodes is $N = 10{,}000$, the number of nodes per cell is 100, and the number of cells is 100. According to Table 1, $t$ is less than 181. Hence, each node needs to perform only 361 64-bit or 128-bit modular multiplications. Similarly, the number of modular multiplications of other polynomial-based schemes is given in Table 3. Obviously, LAKE is more efficient than most conventional symmetric key schemes.

Public key techniques such as RSA and Diffie-Hellman can also achieve key agreement. The basic operation of RSA and Diffie-Hellman is the modular exponentiation of the form $y^x \pmod q$. One modular exponentiation needs $\frac{3}{2}\log_2 q$ $\log_2 q$-bit modular multiplications on average [39]. To guarantee the same level of security, here, $q$ is usually 1,024 bits and $y$ and $x$ are drawn from $\mathbb{Z}_q^*$. Thus, public key-based operations require 1,536 1,024-bit modular multiplications on average. A 1,024-bit modular multiplication is $\left(\frac{1{,}024}{64}\right)^2 = 256$ times more expensive than a 64-bit modular multiplication [15]. Hence, the public key techniques are $256 \times \frac{1{,}536}{361} = 1{,}089$ times more expensive than LAKE if 64-bit symmetric keys are used in LAKE.

## 5 Discussion

LAKE uses a key agreement model derived from our work in [26] where a multivariate polynomial is used to achieve key agreement for large-scale distributed systems. We can also extend LAKE into higher dimensional space and, thus, our work [26] can be used. In this case, each node can be identified by $k$ indices $(n_1, n_2, \ldots, n_k)$ for $k$-dimensional extension and assigned a share of a global $t$-degree $(k+1)$-variate symmetric polynomial. If two nodes have one mismatch in their $k$-tuple IDs, they can calculate a direct key independently. If they have more than one mismatch in their IDs, they can negotiate an indirect key through a secure path consisting of multiple agent nodes. It has been shown in [26] that the minimum polynomial degree can be bounded as

$$t \leq \sqrt[k]{N}\sqrt[k+1]{\frac{k(k+1)!}{2}}, \tag{27}$$

where $N$ is the total number of nodes.

The extensions of PIKE [20] and Hypercube [17] are similar to our work. They also put sensor nodes in a multidimensional grid and predistribute secrets so that each node has pairwise keys with $k(\sqrt[k]{N}-1)$ nodes which are on the $k$ axes passing through it. To guarantee the perfect security of direct keys, each node must store $k(\sqrt[k]{N}-1)$ keys. Therefore, the memory cost of PIKE [20] and Hypercube [17] is higher than ours. In other words, given the same amount of memory resource, our scheme can achieve a higher security level than PIKE [20] and Hypercube [17].

## 6 Conclusion

LAKE uses a $t$-degree trivariate polynomial to facilitate the key establishment between sensor nodes in a two-dimensional space. It can efficiently establish both TLKs and LLKs in sensor networks. Any two nodes can negotiate a TLK with the help of no more than one intermediate node. As for the LLK establishment, LAKE is more secure under the node compromise attack with much less memory cost than conventional schemes. Due to the location-based deployment, LAKE is also energy efficient in that each node has direct keys with almost all neighbors and does not need to consume too much energy on the establishment of indirect keys with neighbors through multihop routing, thus saving significant communication overhead.

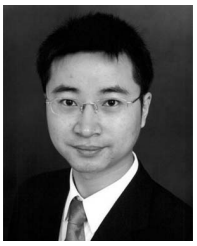
## Acknowledgments

This work was supported in part by the US National Science Foundation under grant CNS-0626881 and under grant ANI-0093241 (CAREER Award).

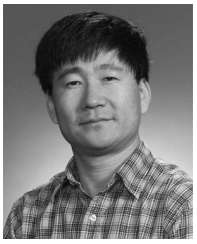
## References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Comm. Magazine*, vol. 40, no. 8, pp. 102-114, Aug. 2002.
[2] J.M. Kahn, R.H. Katz, and K.S.J. Pister, "Next Century Challenges: Mobile Networking for Smart Dust," *Proc. MobiCom*, pp. 217-278, Aug. 1999.
[3] G.J. Pottie and W.J. Kaiser, "Wireless Integrated Network Sensors," *Comm. ACM*, vol. 43, no. 5, pp. 51-58, May 2000.

[4] H.T. Kung and D. Vlah, "Efficient Location Tracking Using Sensor Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC '03),* Mar. 2003.
[5] R. Brooks, P. Ramanathan, and A. Sayeed, "Distributed Target Classification and Tracking in Sensor Networks," *Proc. IEEE,* vol. 91, no. 8, pp. 1163-1171, 2003.
[6] A. Wood and J. Stankovic, "Denial of Service in Sensor Networks," *IEEE Computer Magazine,* vol. 35, no. 10, pp. 54-62, Oct. 2002.
[7] C. Karlof and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures," *Proc. First IEEE Int'l Workshop Sensor Network Protocols and Applications (SNPA '03),* May 2003.
[8] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK: Securing Sensor Networks with Public Key Technology," *Proc. Second ACM Workshop Security of Ad Hoc and Sensor Networks (SASN '04),* Oct. 2004.
[9] D.J. Malan, M. Welsh, and M.D. Smith, "A Public-Key Infrastructure for Key Distribution in Tinyos Based on Elliptic Curve Cryptography," *Proc. First IEEE Int'l Conf. Sensor and Ad Hoc Comm. and Networks (SECON '04),* Oct. 2004.
[10] A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, and D.E. Culler, "SPINS: Security Protocols for Sensor Networks," *Wireless Networks,* vol. 8, pp. 521-534, Sept. 2002.
[11] L. Eschenauer and V. Gligor, "A Key Management Scheme for Distributed Sensor Networks," *Proc. Ninth ACM Conf. Computer and Comm. Security (CCS '02),* Nov. 2002.
[12] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," *Proc. IEEE Symp. Security and Privacy,* pp. 197-213, May 2003.
[13] R.D. Pietro, L.V. Mancini, and A. Mei, "Random Key-Assignment for Secure Wireless Sensor Networks," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03),* Oct. 2003.
[14] M. Ramkumar and N. Memon, "An Efficient Random Key Predistribution Scheme," *Proc. IEEE Global Telecomm. Conf. (Globecom '04),* Dec. 2004.
[15] W. Du, J. Deng, Y.S. Han, and P.K. Varshney, "A Pairwise Key Predistribution Scheme for Wireless Sensor Networks," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03),* Oct. 2003.
[16] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributied Sensor Networks," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03),* Oct. 2003.
[17] D. Liu, P. Ning, and R. Li, "Establishing Pairwise Keys in Distributed Sensor Networks," *ACM Trans. Information and System Security,* vol. 8, no. 1, pp. 41-77, Feb. 2005.
[18] J. Hwang and Y. Kim, "Revisiting Random Key Predistibution Schemes for Wireless Sensor Networks," *Proc. Second ACM Workshop Security of Ad Hoc and Sensor Networks (SASN '04),* Oct. 2004.
[19] R. Wei and J. Wu, "Product Construction of Key Distribution Sschemes for Sensor Networks," *Proc. 11th Int'l Workshop Selected Areas in Cryptography (SAC '04),* Aug. 2004.
[20] H. Chan and A. Perrig, "Pike: Peer Intermediaries for Key Establishment in Sensor Networks," *Proc. INFOCOM,* Mar. 2005.
[21] D. Liu and P. Ning, "Location-Based Pairwise Key Establishments for Relatively Static Sensor Networks," *Proc. ACM Workshop Security of Ad Hoc and Sensor Networks (SASN '03),* Oct. 2003.
[22] W. Du, J. Deng, Y.S. Han, S. Chen, and P.K. Varshney, "A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge," *Proc. INFOCOM,* Mar. 2004.
[23] D. Huang, M. Mehta, D. Medhi, and L. Harn, "Location-Aware Key Management Scheme for Wireless Sensor Networks," *Proc. Second ACM Workshop Security of Ad Hoc and Sensor Networks (SASN '04),* Oct. 2004.
[24] Y. Zhou, Y. Zhang, and Y. Fang, "LLK: A Link-Layer Key Establishment Scheme in Wireless Sensor Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC '05),* Mar. 2005.
[25] Y. Zhou, Y. Zhang, and Y. Fang, "Key Establishment in Sensor Networks Based on Triangle Grid Deployment Model," *Proc. IEEE Military Comm. Conf. (MILCOM '05),* Oct. 2005.
[26] Y. Zhou and Y. Fang, "A Scalable Key Agreement Scheme for Large Scale Networks," *Proc. IEEE Int'l Conf. Networking, Sensing and Control (ICNSC '06),* Apr. 2006.
[27] S. Basagni, K. Herrin, D. Bruschi, and E. Rosti, "Secure Pebblenets," *Proc. MobiHoc,* 2001.
[28] C.J. Mitchell and F.C. Piper, "Key Storage in Secure Networks," *Discrete Applied Math.,* 1995.
[29] M. Dyer, T. Fenner, A. Frieze, and A. Thomason, "On Key Storage in Secure Networks," *J. Cryptology,* 1995.
[30] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-Secure Key Distribution for Dynamic Conferences," *Proc. Conf. Advances in Cryptology (CRYPTO '92),* vol. 740, pp. 471-486, 1992.
[31] S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient Security Mechanism for Large-Scale Distributed Sensor Networks," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03),* Oct. 2003.
[32] J. Deng, C. Hartung, R. Han, and S. Mishra, "A Practical Study of Transitory Master Key Establishment for Wireless Sensor Networks," *Proc. First IEEE Int'l Conf. Security and Privacy for Emerging Areas in Comm. Networks (SecureComm '05),* Sept. 2005.
[33] Y. Zhou, Y. Zhang, and Y. Fang, "Access Control in Wireless Sensor Networks," *Elsevier Ad Hoc Networks J.,* to appear.
[34] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Location-Based Compromise-Tolerant Security Mechanisms for Wireless Sensor Networks," *IEEE J. Selected Areas in Comm.,* vol. 24, no. 2, pp. 247-260, Feb. 2006.
[35] H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbaugh, "Toward Resilient Security in Wireless Sensor Networks," *Proc. MobiHoc,* May 2005.
[36] W. Du, L. Fang, and P. Ning, "LAD: Localization Anomaly Detection for Wireless Sensor Networks," *Proc. 19th Int'l Parallel and Distributed Processing Symp. (IPDPS '05),* Apr. 2005.
[37] R. Anderson and M. Kuhn, "Tamper Resistance—A Cautionary Note," *Proc. Second USENIX Workshop Electronic Commerce,* pp. 1-11, Nov. 1996.
[38] Crossbow Technology, http://www.xbow.com/, 2006.
[39] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography.* CRC Press, 1996.



**Yun Zhou** received the BE degree in electronic information engineering (2000) and the ME degree in communication and information systems (2003) from the Department of Electronic Engineering and Information Science at the University of Science and Technology of China, Hefei, China. He is currently pursuing the PhD degree in the Department of Electrical and Computer Engineering at the University of Florida, Gainesville. His research interests are in the areas of security, cryptography, wireless communications and networking, signal processing, and operating systems. He is a student member of the IEEE.



**Yuguang Fang** received the PhD degree in systems engineering from Case Western Reserve University in January 1994 and the PhD degree in electrical engineering from Boston University in May 1997. He was an assistant professor in the Department of Electrical and Computer Engineering at New Jersey Institute of Technology from July 1998 to May 2000. He then joined the Department of Electrical and Computer Engineering at the University of Florida in May 2000 as an assistant professor and got an early promotion to an associate professor with tenure in August 2003 and a professor in August 2005. He has published more than 150 papers in refereed professional journals and conferences. He received the US National Science Foundation Faculty Early Career Award in 2001 and the US Office of Naval Research Young Investigator Award in 2002. He has served on many editorial boards of technical journals including *IEEE Transactions on Communications*, *IEEE Transactions on Wireless Communications*, *IEEE Transactions on Mobile Computing*, and *ACM Wireless Networks*. He is a senior member of the IEEE.