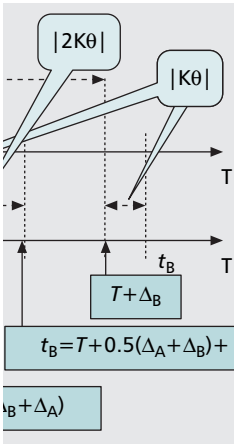
PREDEPLOYED SECURE KEY DISTRIBUTION MECHANISMS IN SENSOR NETWORKS: CURRENT STATE-OF-THE-ART AND A NEW APPROACH USING TIME INFORMATION

JONGMIN JEONG AND ZYGMUNT J. HAAS, CORNELL UNIVERSITY



The authors survey a number of predeployed secure key distribution (PSKD) schemes, which were proposed in the technical literature. They also propose a new time-based predeployed secure key distribution scheme.

ABSTRACT

In this article we survey a number of predeployed secure key distribution (PSKD) schemes proposed in the technical literature. We also propose a new time-based PSKD (TPSKD), which operates under the assumption of loose time synchronization, and discuss the performance of the scheme. Since the TPSKD scheme uses time information, which would typically already be available in sensor nodes, the cost of the scheme's implementation is low.

INTRODUCTION

Sensor networks comprise hundreds to several thousands of sensor nodes. Sensor nodes are low-cost devices with power constraints, are limited in computation and memory capacity, and communicate over a short-range radio interface. Most sensor networks include one or more sink nodes that act as gateways to other network domains. Each sensor node communicates locally with its neighbor nodes to route information to the sink node(s); the network is a peer-to-peer network with multihop routing. Communication traffic within a sensor network falls into two categories: sensor-node-to-sink-node communication for transferring sensor data, and sink-node-to-sensor-node communication for conveying requests and control information.

Because sensor networks are distributed processing (and data acquisition) environments and infrastructureless, sensor networks resemble well-known ad hoc networks. However, this resemblance is superficial only; sensor networks are closed communication environments, and their features differ significantly from those of ad hoc networks. In particular, because of the closed communication environment, it is possible to preconfigure certain mechanisms that are installed before network deployment, without the need to reconfigure these mechanisms again during the lifetime of the sensor network. One such example is the predeployed security mechanism.

Similar to other wireless networks, one of the most significant challenges of sensor networks is provision of secure communication. Due to resource constraints (including communication capabilities limitations) and the physical vulnerability of sensor nodes, which may be subject to physical capture and manipulation by an adversary,[1] some fundamental security functions are difficult to implement. Accordingly, asymmetric key cryptographic functions with their associated relatively large overhead caused by complex mathematical algorithms are impractical. This is so even though asymmetric cryptography is superior to symmetric cryptography in terms of complexity of key management and security strength.

A security scheme relies on both confidential cryptographic keys and known-to-all cryptographic algorithms that use the secure keys as input. Therefore, an implementation of security in a network depends on the ability to securely distribute keys within the network. In the asymmetric cryptographic algorithm, a security binding consists of two different keys; one is a *public key*, which is made widely known, and the other is a *private key*, which is known to one party only. To communicate with a node, the sender uses the widely known public key to encrypt a message into a cipher text, while the recipient uses the private key to decrypt the cipher text. Thus, in the case of asymmetric cryptography, key dis-

The work of Zygmunt J. Haas on this project was supported in part by the National Science Foundation under grants ANI-0329905 and CNS-0626751, and by the MURI Program administered by the Air Force Office of Scientific Research (AFOSR) under contract F49620-02-1-0217.

[1] The vulnerability to physical capture and manipulation is a result of the massive deployment of (often unsupervised) sensor nodes and the low-cost requirement.

1536-1284/08/$25.00 © 2008 IEEE

tribution is simple and easy to implement. In contrast, in the case of symmetric cryptography, a single secure key is used for both encryption and decryption operations. Moreover, this secure key may be used by many (if not all) nodes in the network. Thus, the secure key must be confidentially shared only among the network nodes. Consequently, the implementation of a key distribution in a symmetric cryptography system is significantly more challenging than in an asymmetric cryptography system.

Of course, many symmetric key distribution mechanisms have been proposed in the technical literature; the master key-based, key distribution center (KDC)-based, and public key-based approaches being some such schemes. However, those symmetric key distribution mechanisms are not applicable in hostile, infrastructureless, and resource-constrained sensor networks. Recently a few new approaches for symmetric key distribution that focus on the particular sensor network environment have been studied. For example, earlier work on security in sensor networks, which is based on KDC, is the SPINS protocol [1]. However, more common approaches to key distribution in sensor networks exploit the closed environmental feature of such networks and rely on the predeployed secure key distribution (PSKD) concept. A *predeployed key distribution scheme* is a method to distribute initial security material among a set of users before the network is actually deployed (e.g., at the node manufacturing stage), such that users can compute a common session key for secure communication after the network is deployed.

The main purpose of this article is to study the PSKD mechanism. We analyze and compare several existing PSKD schemes proposed for sensor networks. Additionally, we propose a new approach, the time-based PSKD (TPSKD) scheme, which uses timing information in the sensor nodes for generation of session keys. Since typically sensor nodes already maintain timing information (e.g., for reporting the timing of monitored events), TPSKD does not incur additional implementation cost and reduces the communication costs of session key generation. Furthermore, our scheme has low computing overhead, as it requires only addition and subtraction operations. Finally, unlike in probabilistic approaches, in TPSKD any two sensor nodes can generate a secure session key.

We briefly summarize the basic symmetric key distribution mechanisms and discuss several PSDK schemes designed for sensor networks. We then propose our TPSKD scheme and analyze it. We then conclude the article.

Basic Mechanisms for Symmetric Key Distribution

We classify symmetric key distribution methods into *trusted third-party-based* schemes, *public key-based* schemes, and *a priori information-based* schemes.

Trusted third-party-based schemes: In general, these key distribution schemes rely on a KDC; one notable example is the Kerberos mechanism [2]. In a KDC-based approach, it is assumed that all the nodes in networks trust the KDC, and there already exists a unique and secure shared key between each of the nodes and the KDC. When two nodes need to generate a secure session key, one of the nodes places a request to the KDC to generate such a key. After receiving the session key request, the KDC generates the secure session key and sends it individually to each of two nodes encrypted with the shared key between the node and the KDC. Figure 1a shows the basic operation of a key sharing scheme based on KDC.

Public-key-based schemes: Public key cryptography provides not only a powerful secure communication method, but also an efficient symmetric key distribution method. Hence, many secure applications adopt public-key-based approaches to distribute (establish) symmetric session keys, allowing the use of simpler and less expensive symmetric key cryptography for the actual secure communication of information. A well-known key distribution scheme based on public key cryptographic is the Diffie-Hellman (DH) exponential key exchange protocol [3]. To generate a secret symmetric shared key, each of two entities randomly selects a private value using a common nonsecret, sufficiently large prime number p. Then, using a known-to-all generator value g, each of the two entities calculates the exponent function of g raised to the power of its private value and sends in the clear the result to the other entity. Each entity then raises the value received from the other entity to the power of its private value, creating a shared secret session key. This DH procedure is depicted in Fig. 1b.

A priori information-based schemes: In these schemes a key setup server distributes specific information (seed) to all nodes before the nodes need to establish shared secret session keys. The seed is then used to generate the session keys. For example, when the seed is implanted in the nodes prior to the nodes′ deployment (e.g., at manufacturing time), the a priori information-based scheme is called the *predeployed key distribution scheme*. Thus, when nodes execute the algorithm to generate a shared secure session key, the nodes are already in possession of the seed information. In general, in the a priori information-based scheme, the way to distribute this seed information depends on the network environment. Figure 1c shows an example of the a priori information-based scheme.

Even though the KDC-based and public key-based key distribution schemes are well known and useful key distribution methods for securing communication protocols, those protocols do not match the specific attributes of sensor networks. Those attributes include a closed,[2] resource-constrained, infrastructur-less, and multihop communication environment. Among the disadvantages of KDC-based schemes is the fact that a security binding would be necessary between one node (the KDC) and all other nodes in the network, which necessitates the KDC storing a large number of shared keys, cor-

Even though the KDC-based and the public key-based key distribution schemes are well-known and useful key distribution methods for securing communication protocols, those protocols do not match the specific attributes of sensor networks.

[2] *By "closed communication environment" we mean that the set of network nodes does not change.*

Most predeployed threshold-based key distributions schemes execute the following procedures: key predistribution (initialization); direct (link) key establishment (key setup); and indirect (path) key establishment.

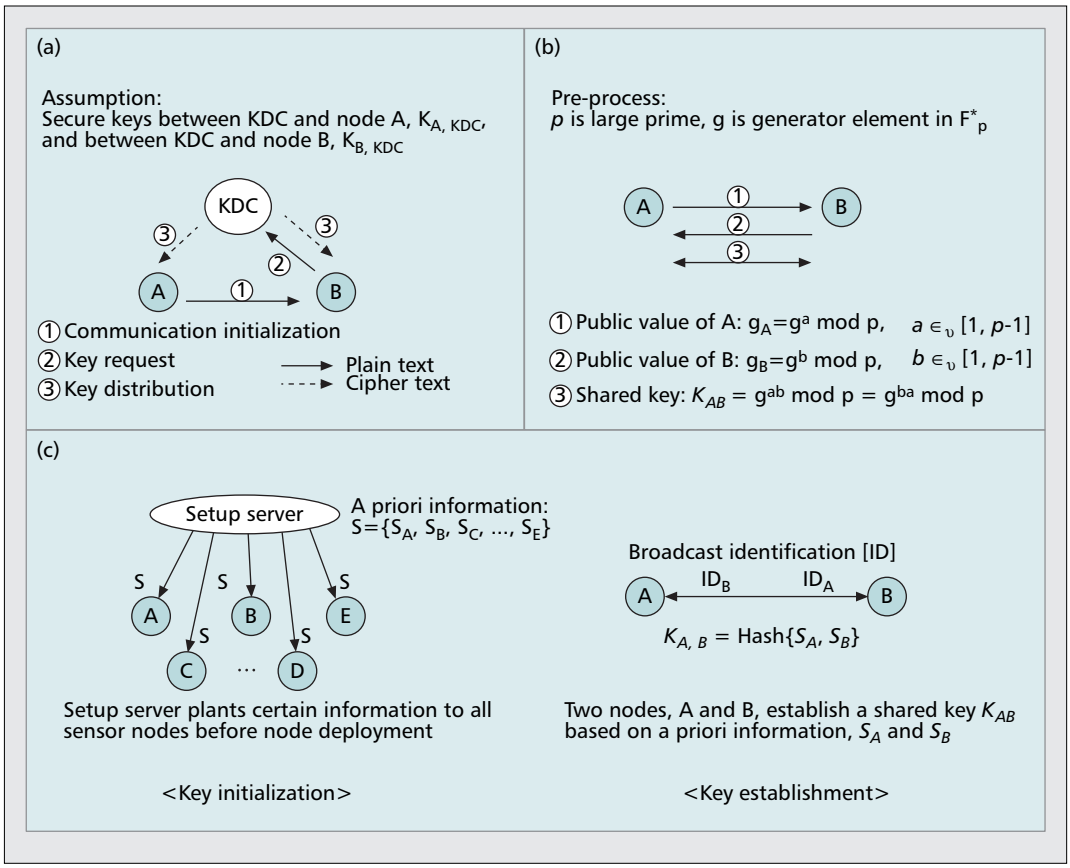


Figure 1. *Symmetric key distribution mechanisms: a) KDC-based scheme; b) public key cryptography-based scheme; c) a priori information-based scheme.*

responding to the massive number of sensor nodes. In the case of public-key-based schemes, due to their strong reliance on public key cryptography, the processing overhead may be prohibitively large. On the other hand, a priori information-based schemes are much better suited to the particular features of sensor networks.

In this article we concentrate on a priori information-based schemes based on predeployed key distribution as a means to generate symmetric secure session keys between nodes in sensor networks.

Predeployed Threshold-Based Key Distribution

A predeployed threshold-based key distribution scheme is a method to distribute pieces of the secret seed among a group of users. A parameter, S_{min}, is defined, where each subset of the group of size S_{min} or larger can compute a common session key for secure communication. In this section we consider current symmetric key distribution schemes based on predeployed threshold-based key distribution for sensor networks.

Basic Operations of Key Predistribution Procedures

Most predeployed threshold-based key distributions schemes execute the following procedures:

- **Key predistribution (initialization):** This phase specifies how to predistribute the key material to each sensor node (e.g., randomly selecting a key ring, using a symmetric matrix, or using a symmetric polynomial).
- **Direct (link) key establishment (key setup):** This phase specifies how to directly establish a pair-wise shared key between two adjacent sensor nodes.
- **Indirect (path) key establishment:** Given two (not neighboring) nodes, the source and destination nodes, this phase specifies how to find a sequence of nodes between the two nodes to establish a session key.

Random Pool-Based (Probabilistic Key Distribution) Schemes

Basic Random Key Pre-Distribution — Eschenauer *et al.* [4] proposed the *basic random key predistribution scheme*. The scheme's operation is shown in Fig. 2a. At the key initialization phase, a random key pool is selected from the key space, and before deployment each sensor node receives from the key pool a random subset of keys, which constitutes a key ring. At the key setup phase, each node broadcasts identification of its own keys to discover a shared key with the adjacent nodes. Any two nodes able to find at least one common key can use that key as their shared secret to initiate communication. After the key setup stage, a graph of secure links is formed that consists of all the links between adjacent nodes who share at least one key. If the graph is

connected, a path can be found from the source node to the destination node. Then, source node can generate a session key and send it securely via the link key of each individual link to the destination node.

Enhanced Random Key Pre-Distribution — Chan *et al.* [5] proposed three new methods of modified random key predistribution to address the bootstrapping problem. The bootstrapping problem (known to be challenging) refers to the procedure for initial establishment of a secure communication infrastructure from a collection of sensor nodes. Those nodes might have been preinitialized with some secret information, but had no prior direct contact with each other [5]. First, the q-*composite random key predistribution scheme*, illustrated in Fig. 2b, requires q common keys instead of just one, as in the basic random key predistribution scheme. By increasing the amount of key overlap required for key setup, the scheme increases the resilience of the network against node capture. However, to preserve the probability of two nodes sharing a sufficient number of keys to establish a secure link, it is necessary to reduce the size of the key pool. This allows the attacker to gain a larger sample of the key pool by breaking fewer nodes. A new communication link key K is generated by a *hash* function of shared keys (such that $q' \geq q$ and $K = hash\ (K_1||K_2||\ldots||K_{q'})$. Second, a *multipath key reinforcement scheme* is used to update the communication key to a random value over multiple independent paths after a key setup procedure has been completed for each path. This scheme assumes that the source node is able to discover disjoint paths to the destination node. Assume that there are i paths between the source and destination nodes. Then the source node generates i random values, R_i, and sends each random value along a different path to the destination. The new session key can be computed by both source and destination using $K' = K \oplus R_1 \oplus R_2, \ldots, \oplus R_i$. Third, a *random pair-wise key scheme* provides node-to-node authentication. Because keys can be issued multiple times out of a key pool, another node (e.g., an adversary node) can also hold the same secret key in its key ring. Two nodes sharing a session key, K, cannot ascertain that they have established secure communication between them, since secure communication requires authentication of the communicating parties. In the initialization phase, to generate the key pool, nodes' identities are randomly matched, so a key corresponds to a pair of node identities. In the key setup phase each node broadcasts its node identity, and by searching for the identities of the other nodes in its key ring, a node can determine if it shares a common pair-wise key with its neighbors.

Symmetric Matrix-Based Schemes

Basic Matrix-Based Key Predistribution — Blom [6] proposed a key predistribution scheme that allows any pair of nodes to find a secret pairwise key between them. Compared to the pairwise key predistribution scheme in which each node maintains $n - 1$ pair-wise keys (where n is the number of nodes in a network), Blom's scheme uses only $\lambda + 1$ memory spaces, where λ is much smaller than n. However, unlike the $(n - 1)$ pair-wise key scheme, Blom's scheme is not perfectly resilient against node capture; instead it has the following λ-secure property: As long as an adversary compromises no more than λ nodes, the uncompromised nodes are secure. But when an adversary compromises more than λ nodes, all pairwise keys of the entire network are compromised. During the predeployment phase, the setup server first constructs a $(\lambda + 1)$ by n matrix G, over finite field GF(q), where n is the size of the networks. The matrix G is made widely known, and any sensor node, including an adversary node, has access to the content of G. Then the setup server creates a random $(\lambda + 1)$ by $\times\ (\lambda + 1)$ symmetric matrix D over GF(q), and computes an n by $(\lambda + 1)$ matrix $A = (D \cdot G)^T$, where $(D \cdot G)^T$ is the transpose of $D \cdot G$. The matrix D is kept secret and should not be disclosed to any sensor node [7]. Then, each node i stores the i-th row of the matrix A. Because D is a symmetric matrix, $K = A \cdot G$ is symmetric as well, and $K_{ij} = K_{ji}$ serves as a pairwise key between nodes i and j. Figure 3a illustrates a link key establishment procedure based on a basic matrix-based key predistribution scheme.

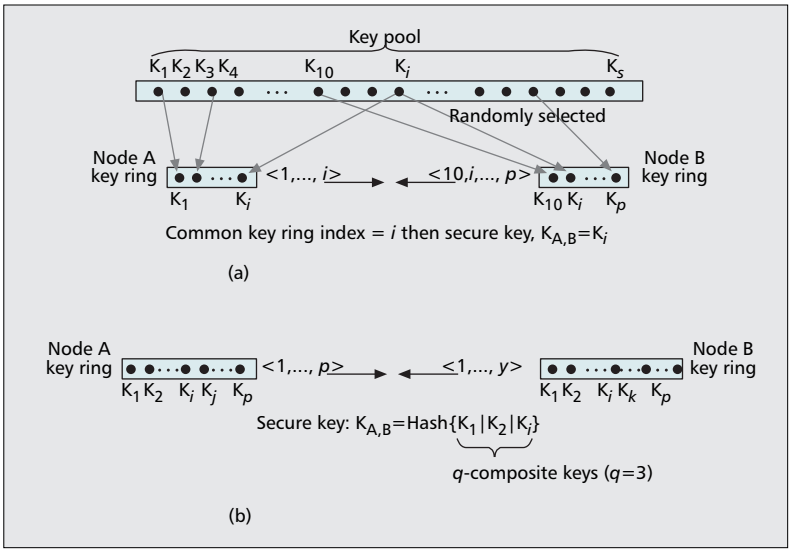


Figure 2. *The random pool-based schemes: a) basic random pool-based scheme; b)* q-*composite random pool-based scheme.*

Matrix-Based Multiple-Space Key Predistribution — Du *et al.* [7] proposed a *multiple key spaces predistribution scheme* based on the Blom's scheme. The generation of the G matrix is identical to Blom's scheme. The difference is that the scheme in [7] generates ω symmetric matrices $D_1, \ldots, D_\omega$ of size $(\lambda + 1) \times (\lambda + 1)$. Each tuple $S_i = (D_i\ G)$, $1 \leq i \leq \omega$, is called a key space. The matrix $A_i = (D_i \cdot G)^T$ is computed, and for each node, τ distinct key spaces are selected from the ω key spaces. For each space S_i selected by node j, the jth row of A_i is stored at node j. After deployment, each node discovers whether it shares any space with its neighbors by broadcasting a message containing the node identity, the indices of the spaces, and the seed of the column of G. Two nodes that find out that they have a com-

In the key setup phase, each node broadcasts its node identity and by searching for the identities of the other nodes in its key ring, a node can determine if it shares a common pair-wise key with its neighbors.

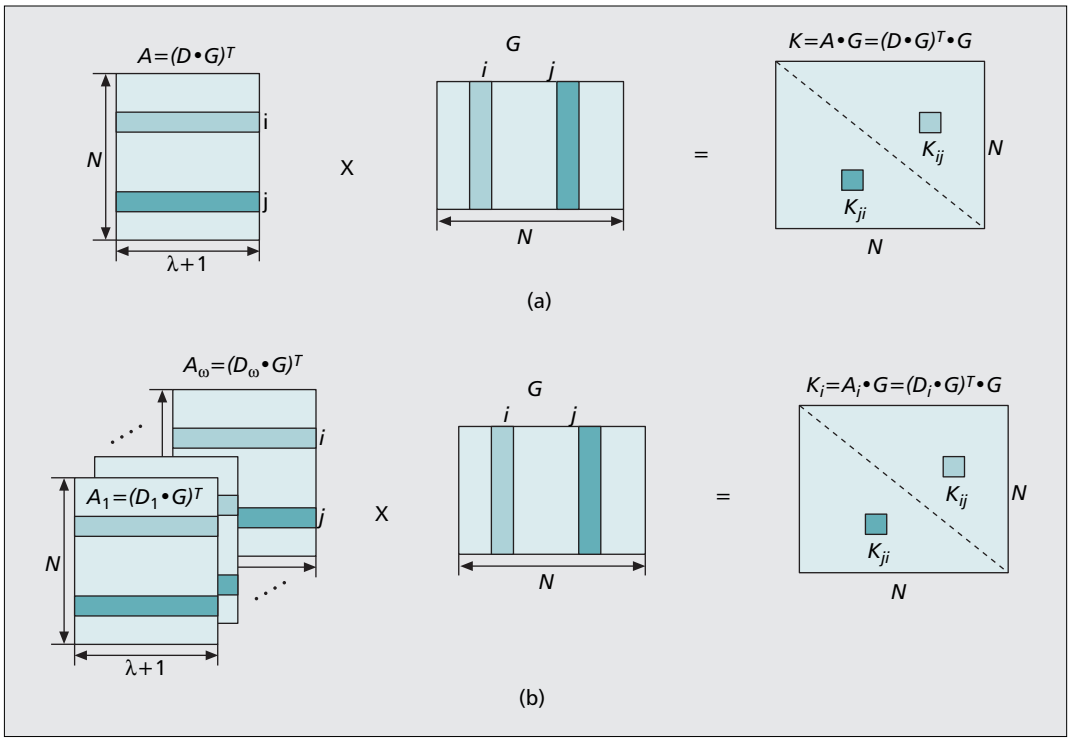


Figure 3. *Symmetric matrix-based key predistribution schemes: a) basic symmetric matrix-based scheme; b) multiple-space matrix-based scheme.*

mon space can compute their pairwise secret key using the Blom's scheme. Figure 3b shows a link-key establishment procedure based on the multiple-space key predistribution scheme.

Location-Based Scheme

Closest Pair-Wise Key Predistribution — Liu *et al.* [8] proposed a *closest pair-wise key predistribution scheme*. In the initialization (predistribution) phase, for each sensor i, the key setup server first discovers a set $\boldsymbol{S}$ of $\boldsymbol{c}$ sensors whose expected deployment locations are closest to the expected deployment location of sensor i. For each sensor j, the key setup sever randomly generates a unique pair-wise key K_{ij} and distributes (i, K_{ij}) and (j, K_{ji}) to sensors i and j, respectively. After the deployment of the sensor nodes, if two sensors i and j want to set up a pair-wise key to secure the communication between them, they only need to check whether they have a predeployed pair-wise key with the other party. The process of indirect key establishment is similar to some other mechanisms, such as the random pool mechanism. In the extended version of the scheme, which is proposed to reduce the storage requirements at no extra communication overhead, the setup sever allocates a master key K_i to each sensor i. The server then computes a set of pair-wise keys between node i and every one of its neighbors, $j \in S$, using a pseudo-random function,[3] $K_{ij} = PRF_{K_j}(i)$, and distributes this key set to sensor node i. To obtain the link key between node j and its neighbor node i, node j computes K_{ij}. Consequently, not all nodes need to store all the link keys between them and their neighbors.

Polynomial-Based Schemes

Basic Polynomial-Based Key Predistribution — Blundo *et al.* [9] proposed a polynomial-based key distribution scheme for dynamic conferencing, which could be used for sensor networks. Even though this work was proposed to generate a group key of t nodes out of n nodes, for simplicity, we discuss here a specific case of pair-wise key distribution, $t = 2$. The setup server randomly generates a bivariate k degree symmetric polynomial $f(x, y) = f(y, x)$ over finite field GF(q), $q > n$. For each sensor i, a key setup server computes a polynomial share of $f(x, y)$, $f(i, y)$. For any two sensor nodes i and j, i can compute the common key $f(i, j)$ by evaluating $f(i, y)$ at point j, while j can compute the same key $f(j,i) = f(i, j)$ by evaluating $f(j, y)$ at point i.

Polynomial Pool-Based Key Predistribution — Liu *et al.* [10] proposed the *polynomial pool-based key predistribution* scheme, which uses multiple random bivariate polynomials. Its idea is inspired by the random pool model [4, 5]. During initialization, the setup server randomly generates a set F of bivariate k-degree polynomials over the finite field GF(q). For each sensor node i, the setup server picks a subset of polynomials $F_i \subset F$, and assigns the polynomial shares of these polynomials to node i. If both sensors have polynomial shares of the same bivariate polynomial, they can establish a pair-wise key directly using the basic polynomial-based key predistribution scheme.

Grid and Polynomial-Based Key Predistribution — Liu *et al.* [10] proposed the grid-based key predistribu-

[3] *Liu* et al. *do not define how two parameters* K_j *and* i *are used in the pseudo-random function, PRF( ).*

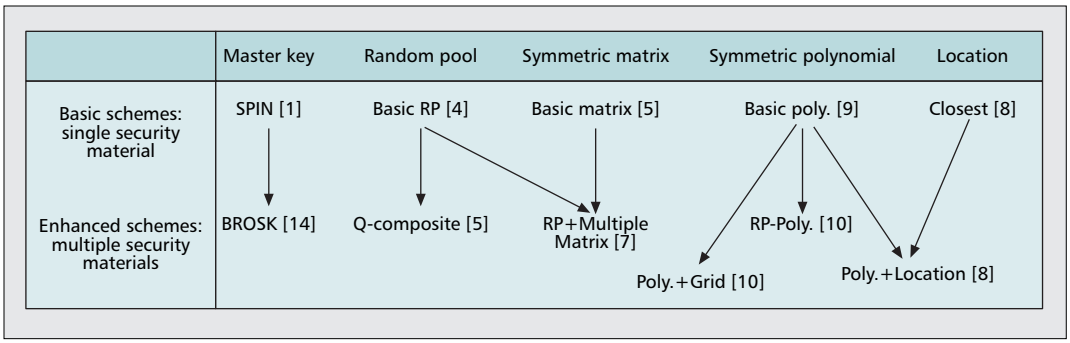


Figure 4. *Summarization of pair-wise session key schemes.*

tion scheme based on the polynomial scheme. Provided that a sensor network has at most n sensor nodes, the grid-based key predistribution scheme constructs an m by m grid from a set of $2m$ polynomials $\{f_i^c(x, y), f_i^r(x, y)\}, i = 0, \ldots, m - 1$, where $m = \sqrt{n}$. Each row i in the grid is associated with a polynomial $f_i^r(x, y)$, and each column j is associated with a polynomial $f_j^c(x, y)$. The setup server assigns each sensor in the network to a unique intersection in this grid such that ID = (i, j) and distributes $\{ID, f_i^c(j, x), f_j^r(i, x)\}$. During the key setup phase, to establish a pair-wise key with node j, node i checks whether $c_i = c_j$ or $r_i = r_j$. If $c_i = c_j$, or $r_i = r_j$, both nodes i and j have polynomial shares of $f_{c_i}^c(x, y)$ or $f_{r_i}^r(x, y)$, respectively, and the nodes can use polynomial-based key predistribution to establish a pair-wise key directly.

Polynomial and Location Information-Based Key Predistribution — Liu *et al.* [8] proposed a key predistribution scheme by using bivariate polynomials combined with the closest pair-wise keys scheme based on location information [8]. Instead of assigning each sensor set of pair-wise keys, the scheme distributes to each sensor a set of polynomial shares that belong to the closest set in which this sensor is expected to be located. After deployment of polynomial shares, sensor nodes first need to identify a shared bivariate polynomial. If they can find at least one such polynomial, a common pair-wise key can be established directly using the basic polynomial-based key predistribution scheme [9].

Figure 4 summarizes the pair-wise session-key establishment schemes based on predeployed security materials. As discussed above, the basic approaches can be classified into five basic types, and enhancements of those approaches have been proposed, including schemes based on combinations of the five basic types.

Group-Wise Key Predistribution

Polynomial-Based Key Predistribution — Blundo *et al.* [9] proposed several schemes that allow any group of t parties to compute a common key while being secure against collusion between some of them. The setup server randomly generates a symmetric polynomial in t variables of degree k with coefficients over GF(q), $q > n$. The server provides each user u_i the polynomial $f_i(x_2, \ldots, x_t) = g(i, x_2, \ldots, x_t)$, which is the polynomial obtained by evaluating $g(x_1, \ldots, x_t)$ at $x_1 = i$. If users $u_1, u_2, \ldots u_t$ want to set up a group key, each user u_i evaluates f_i $(x_2, \ldots, x_t)$ at $(x_2, \ldots, x_t) = (1, \ldots, i - 1, i + 1, \ldots, t)$. The group key is equal to $K_1,\ldots, t = g(1, 2, \ldots, t)$. These schemes focus on saving communication costs without placing memory constraints on group members. When $t = 2$, one of these schemes is a special case of Blom's scheme.

Group Key Predistribution — Liu *et al.* [11] proposed *group key predistribution* by observing that long distance peer-to-peer secure communication between sensor nodes is rare. This scheme does not require knowledge of a sensor's location, unlike the other location-based predistribution schemes [8]. Instead, the scheme assumes that sensor nodes are deployed in groups, and nodes in the same group are located close to each other. The pair-wise in-group key predistribution, by which each sensor node in the same deployment group establishes pair-wise keys with other nodes, is based on a set of keying materials such as random keys, polynomial-based, or matrix-based. To handle pair-wise key establishment between sensor nodes in different deployment groups, a cross-group key predistribution process is performed, which enables selected sensor nodes in different deployment groups to establish pair-wise keys to bridge different deployment groups. If there are n equal size deployment groups with m sensor nodes in each group, a setup server constructs in-group domains G_i, $(1 \le i \le n)$ and cross-group domains G_i', $(1 \le i \le m)$. Basically, each deployment group G_i contains sensor nodes with IDs of $\{(i - 1)m + j\}_{j=1,\ldots,m}$, while each cross-group G_i' contains sensor nodes with IDs of $\{i + (j - 1)m\}_{j=1,\ldots,m}$.

Time Information-Based PSKD

We introduce a new predeployed key distribution mechanism based on time information to generate secret link or session keys. We utilize time information as predeployed security material. First, we list the following assumptions used in the design of our scheme:

- **Assumption #1: Loose time synchronization:** Even though clocks of sensor nodes would typically not be tightly synchronized (due to the low-cost hardware of the nodes), loose time synchronization can be assumed in practice. Moreover, based on the hardware design and implementation, an estimation of the *maximum drift per unit time* (say, θ [s/year]) of the sensor node clocks can be calculated.

After deployment of polynomial shares, sensor nodes first need to identify a shared bivariate polynomial. If they can find at least one such polynomial, common pair-wise key can be established directly using the basic polynomial-based key pre-distribution scheme.

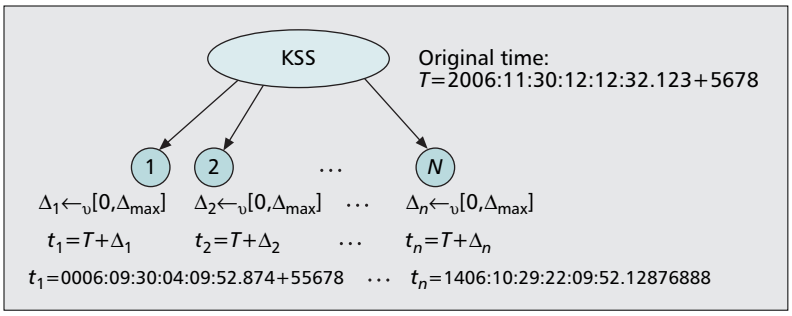


Figure 5. *Key initialization in the TPSKD scheme.*

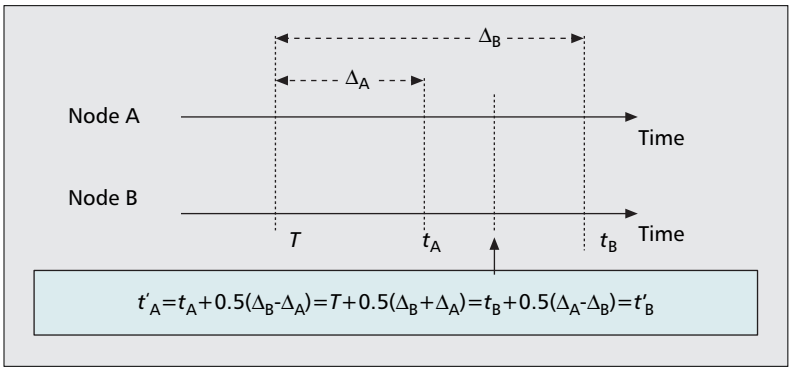


Figure 6. *Clock resynchronization due to intentionally introduced delays, Δs.*

Thus, if at the manufacturing stage all the clocks are set to the same value, the difference between the clocks of any two nodes in the network after k years will be at most $2k\theta$[s]. Indeed, time synchronization is one of the fundamental requirements in many applications of sensor networks[4] (e.g., [12, 13]). Hence, loose time synchronization is a practical assumption.

- **Assumption #2: Physical node security:** We assume that the nodes are equipped with tamper-proof memory. In essence, we assume that nodes are able to store some secret values and that a physical capture of the nodes will not compromise the stored secret values (e.g., tampering with the node erases its memory content). The tamper-proof memory assumption has been used in the technical literature before (e.g., [16, 17]).

The Principles of the TPSKD Scheme

We assume that node i in the network maintains the time t_i of its clock and that the maximum drift of a network node's clock is θ [s/year]. Furthermore, we assume that at the manufacturing stage a common time value T is used to set up all the nodes.[5] The parameter T is used in the key initialization stage only and is not stored in the network nodes.

As in other PSKD schemes, the TPSKD scheme consists of three stages: key initialization, which specifies the time information predistribution; pair-wise link key setup, which specifies a symmetric secure key negotiation procedure with neighbor nodes; and path key setup, which specifies an establishment of a path key between the source and destination nodes using already established secure pair-wise link keys.

Key Initialization Procedure — At the predeployment stage, the key setup server (KSS) randomly generates a *time offset* Δ_i for each node i according to a uniform random distribution $\Delta_i \sim U[0, \Delta_{max}]$. Then the KSS produces the initial time value of the node i clock, t_i, using a function G, $t_i = G(\Delta_i, T)$. Once the initial time values are loaded into the clocks of the nodes, the clocks continue to advance in time. An operation of the algorithm G is run in the KSS only and is secret; that is, the time values, t_i, are kept secret in the network nodes. A basic instance of the algorithm G is the addition operation, which we use here for exemplary purposes. The key initiation procedure is depicted in Fig. 5.

Pair-Wise Link Key Establishment — The goal of the TPSKD is to arrive at the same value of a link key at two communicating nodes, A and B. This is achieved by using the values of the time clocks, t_A and t_B, at the two nodes. Of course, the problem is that, in general, $t_A \neq t_B$ because of three reasons:

1 The clocks were intentionally offset from T by random time offsets, Δ_A and Δ_B, at the manufacturing stage.
2 The clocks exhibit drift relative to T, the maximum drift being θ [s/year].
3 The time at which the selection of the link key commences in the two nodes is different; the difference accounts for the propagation time between the two nodes and different processing times at the two nodes.

We show how to synchronize the clock values, t_A and t_B, in spite of the above three sources of mis-synchronization. The synchronization is achieved at the expense of some degree of clock granularity. But since in the TPSKD scheme we do not use the clocks for timing purposes, more coarsely synchronized clock values suffice.

The TPSKD scheme addresses the intentionally introduced offset from T by random time offsets, Δ_A and Δ_B, by exchanging these offsets between the two nodes. Thus, if $t_A = T + \Delta_A$ and $t_B = T + \Delta_B$,

$$t_A + 0.5\,(\Delta_B - \Delta_A) = T + 0.5\,(\Delta_B + \Delta_A) = t_B + 0.5\,(\Delta_A - \Delta_B).$$

Consequently, after exchanging the offsets, Δ_A and Δ_B, node A calculates $t'_A = t_A + 0.5(\Delta_B - \Delta_A)$ and node B calculates $t'_B = t_B + 0.5\,(\Delta_A - \Delta_B)$. If mis-synchronization causes 2 and 3 were absent, the two nodes would now share the same value, $t'_A = t'_B$. We refer to this shared value as the link key. Note that even though Δ_A and Δ_B are exchanged in the clear, the values of t'_A and t'_B are still kept secret, since the values t_A and t_B are maintained secret at their respective nodes. This scenario, where the only source of mis-synchronization is the intentionally introduced offset, is shown in Fig. 6.

To address the drift of the clocks over time

[4] *Such as timing monitored events.*

[5] *The parameter can assume any initial value, which does not need to be related to the actual time.*

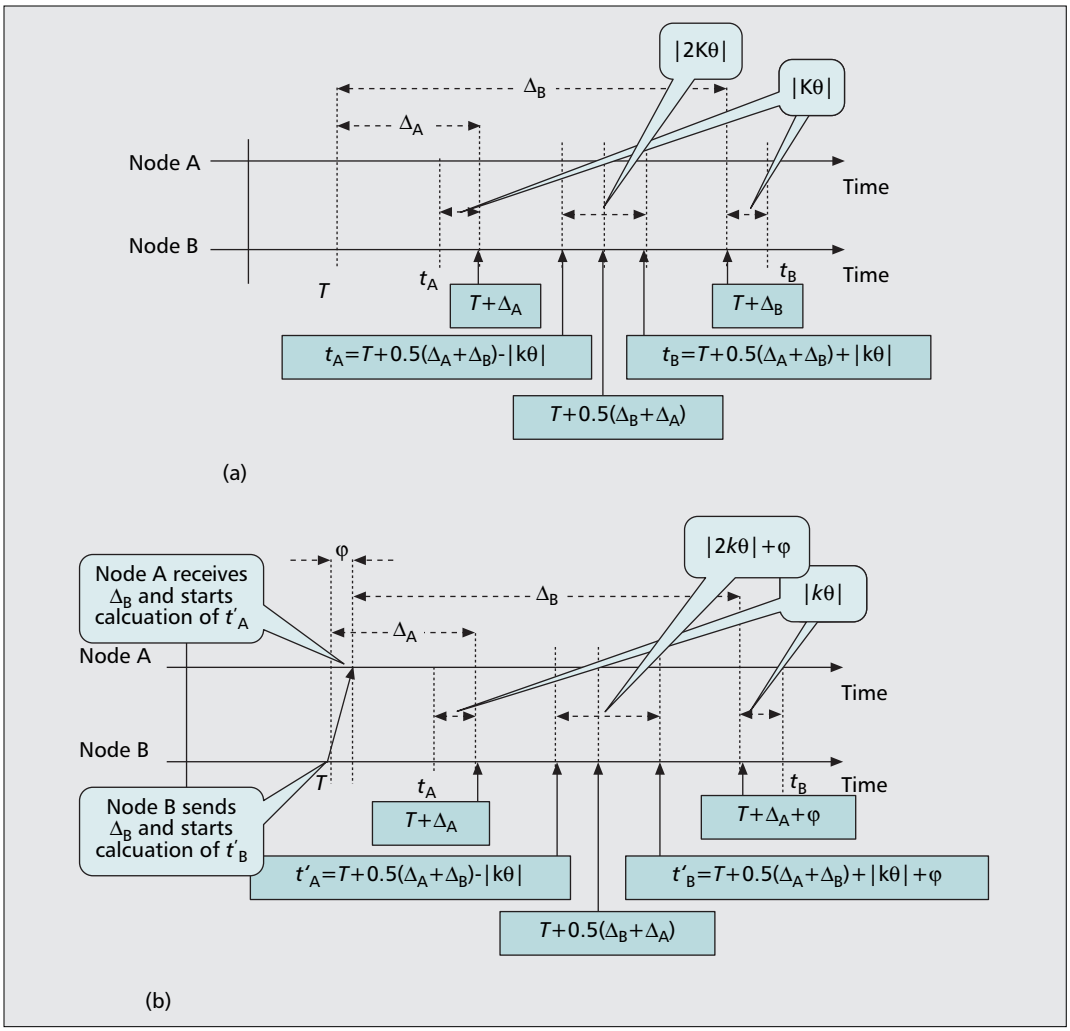


Figure 7. *Clock resynchronization: a) due to clock drifts, with maximum drift* Θ(k)*, and due to intentionally introduced delays,* Δ*s; b) due to clock drifts, with maximum drift* Θ(k)*, due to intentionally introduced delays,* Δ*s, and due to propagation delay,* φ.

(cause 2), we calculate the maximum mis-synchronization of the values t'_A and t'_B over time. We now prove that since the absolute drift of each of the two clocks is no more than θ [s/year], k years after the manufacturing of the nodes, the maximum drift between the two *link-keys* is no more than $\Theta(k) = |2k\theta|$.

The largest difference (worst case scenario) between t'_A and t'_B in the presence of clocks drifts occurs when the drift of the clock with the smaller Δ_A is negative and the drift of the clock with the larger Δ_B is positive. Assuming, without loss of generality, that $\Delta_A \le \Delta_B$, then

$$t_A = T + \Delta_A - |k\theta|;\ t_B = T + \Delta_B + |k\theta|;$$
$$t'_A = t_A + 0.5(\Delta_B - \Delta_A) = T + 0.5(\Delta_B + \Delta_A) - |k\theta|;$$
$$t'_B = t_B + 0.5(\Delta_A - \Delta_B) = T + 0.5(\Delta_A + \Delta_B) + |k\theta|.$$

Thus, $\max(|t'_B - t'_A|) = |2k\theta| = \Theta(k)$; that is, the maximal difference between the values of the link-keys of the two nodes is $\Theta(k)$.This is shown in Fig. 7a.

We note that the value of $\Theta(k)$ could be bound by assuming k equal to the maximum lifetime of the network, or it could be evaluated (e.g., computed) every so often (e.g., every several months). We will see how the value of $\Theta(k)$ is used to correct the t'_A and t'_B after we consider the effect of cause 3.

Finally, the third cause of clock mis-synchronization, due to the maximal propagation delay of φ between the two nodes, is shown in Fig. 7b. We assume that node A initiates the selection of the *link-key* by sending a request to node B. (In fact, node A's request can already contain the value of Δ_A). Node B, upon receiving the request, proceeds with transmitting its value Δ_B to node A and starts with calculation of the value t'_B. Node A receives the Δ_B after time φ, at which time node A starts the calculation of t'_A.

We calculate now the maximum mis-synchronization of the values t'_A and t'_B:

$$t_A = T + \Delta_A - |k\theta|;\ t_B = T + \Delta_B + |k\theta| + \varphi;$$
$$t'_A = t_A + 0.5\,(\Delta_B - \Delta_A) = T + 0.5\,(\Delta_B + \Delta_A) - |k\theta|;$$
$$t'_B = t_B + 0.5\,(\Delta_A - \Delta_B) = T + 0.5\,(\Delta_A + \Delta_B) + |k\theta| + \varphi.$$

Thus, $\max(|t'_B - t'_A|) = |2k\theta| + \varphi = \Theta(k) +$

Provided that link keys are securely established and the network represents a connected communication graph, the operation to generate a path key is based on the same procedure as generation of link keys.

The resource-constrains and the closed-environment features of sensor networks dictate the selection of symmetric-key cryptography as the cryptographic mechanism. Therefore, securing a symmetric key distribution is an essential requirement.

φ. In practice, φ is estimated based on the maximal distance of two nodes which can still communicate in the network. Furthermore, if there may be difference in processing time at the two nodes, this difference should also be included in φ.

We now proceed with the explanation of how the two link-keys, t'_A and t'_B, could be synchronized at the two nodes when we know that the maximum difference between these two values is $\Theta(k) + \varphi$. We assume that clocks t_A and t_B are in binary representation. Then this maximal difference is expressed by a binary number of $1 + \lfloor \log_2 (\Theta(k) + \varphi) \rfloor \equiv \tau$ bits. Thus, if we truncate t'_A and t'_B τ bits from the right, values t'_A and t'_B should not differ by more than 1. As the first example, consider $t'_A = 1101101\mathbf{\underline{000}}$ and $\Theta(k) + \varphi = 111$; then $\tau = 3$, and $t'_B \le 1101101\mathbf{\underline{111}}$. Now, truncating t'_A and t'_B 3 bits from the right yields the same value of $\lfloor t'_B \rfloor_3 = \lfloor t'_A \rfloor_3 = 1101101$, where we label $\lfloor x \rfloor_\tau$ as the value of x truncated τ bits from the right. As our second example, consider that $t'_A = 1101101\mathbf{\underline{001}}$ and $\Theta(k) + \varphi = 111$; then $\tau = 3$, and $t'_B \le 1101110\mathbf{\underline{000}}$. Assuming that the drift is, indeed, maximal and equals 111, truncating t'_A and t'_B 3 bits from the right yields $\lfloor t'_A \rfloor_3 = 1101101$, while $\lfloor t'_B \rfloor_3 = 1101110$ (i.e., the difference of 1). The difference occurs when the truncated value of the larger clock creates carry-over into the bits left to the τ rightmost bits.

First we note that both of the nodes can determine when there is a possibility for the carry-over to occur: when either $\lfloor t'_X \rfloor_\tau \ne \lfloor t'_X (\Theta(k) + \varphi) \rfloor_\tau$ or $\lfloor t'_X \rfloor_\tau \ne \lfloor t'_X (\Theta(k) - \varphi) \rfloor_\tau$. The probability of occurrence of this condition is minimized when τ increases. Thus, setting $\tau >> 1 + \lfloor \log_2(\Theta(k) + \varphi) \rfloor$ will ensure that nearly always the two nodes can find a shared link key on the first attempt of the above procedure. Another option to detect that the carry-over event occurred is for both of the nodes to exchange (in the clear) the least significant bits of their truncated values; if unequal, the nodes declare that the carry-over condition occurred. When the above condition does occur, there are different options of how to proceed. One way is for the nodes to wait a short while and repeat the procedure of link key establishment. Another option when the condition occurs is to use the "round" operation, rather than the "truncate" operation.[6]

Path (Session) Key Establishment — Provided that link-keys are securely established and the network represents a connected communication graph, the operation to generate a path key is based on the same procedure as generation of link-keys, similar to the existing pair-wise key predistribution mechanism operating on a link-by-link basis. The path keys are negotiated between nodes more than one hop away, using already established link keys of the constituent links of the path.

[6] *Note that, in general, using the "round" operation rather than the "truncate" operation results in the same problem, also at different values of* t_A *and* t_B. *However, when the carry-over event occurs with the "truncate" operation, the "round" operation will avoid the problem if the value of* τ *is properly set.*

Summary of the Features of the TPSKD Scheme

- **Memory requirement:** The memory requirements of the TPSKD scheme are minimal; node i needs to store essentially only the values of t_i, Δ_i, and t.
- **Communication:** Similarly, to establish a session key, the two nodes need only to exchange the values of their Δs. Thus, the communication requirements of the TPSKD scheme are negligible.
- **Deterministic link key connectivity:** As long as two nodes in the network can exchange their Δs, the TPSKD scheme is always able to generate a common secret key within the two nodes. Thus, unlike the probabilistic schemes, such as the random pool scheme, the TPSKD is deterministic in its ability to secure link-level connection.
- **Key resilience:** Because of the large possible space of the time values t_i, the probability that an adversary can guess a link-,key of two non-compromised nodes is extremely small.
- **Key refreshment:** Because the value of the session key depends on the values of the continually advancing clocks in the two nodes, the session keys are different each time they are generated. Therefore, since two nodes regenerate the session key with new time information, the session keys are frequently refreshed.

Conclusion

The resource-constrained and closed environment features of sensor networks dictate the selection of symmetric key cryptography as the cryptographic mechanism. Therefore, securing symmetric key distribution is an essential requirement to realize secure communication in sensor networks. Because the closed environment makes it possible to preprogram the sensor nodes in advance of their deployment, it is natural to design the key distribution mechanism of sensor networks based on predeployed information.

In this article we survey several predeployed key distribution mechanisms and propose a new predeployed key distribution scheme (TPSKD) based on time information, which assumes loose time synchronization among network nodes. Because the TPSKD scheme uses time information as key material and time is typically available in sensor nodes, the TPSKD scheme does not require additional hardware. Furthermore, the communication costs of the scheme are also minimal.

References

[1] A. Perrig *et al.*, "SPINS: Security Protocols for Sensor Networks," *Wireless Networks 8*, 2002, pp. 521–34.
[2] B. C. Neuman and T. Ts'o, "Kerberos: An Authentication Service for Computer Networks," *IEEE Commun. Mag.*, vol. 32, no. 9), Sept. 1994, pp. 33–38.
[3] W. Diffie and M. E. Hellman, "New Directions in Cryptography," *IEEE Trans. Info. Theory*, vol. IT-22, no. 6, 1976, pp. 644–44.
[4] L. Eschenauer and V. D. Gligor, "Key-Management Scheme for Distributed Sensor Networks," *9th ACM Conf. Comp. and Commun. Sec.*, 2002.
[5] H. Chan, A. Perrig and D. Song, "Random Key Predistribution Schemes for Sensor Networks," *IEEE Symp. Research in Sec. and Privacy*, 2003.
[6] R. Blom, "An Optimal Class of Symmetric Key Generation Systems," *Eurocrypt '84*.
[7] W. Du *et al.*, "A Pairwise Key Predistribution Scheme for

Wireless Sensor Networks," *Proc. CCS '03*, Oct. 30, 2003.
[8] D. Liu and P. Ning, "Location-based Pairwise Key Establishments for Static Sensor Networks," *1st ACM Wksp. Security of Ad Hoc and Sensor Networks*, 2003.
[9] C. Blundo *et al.*, "Perfectly-Secure Key Distribution for Dynamic Conference," *Crypto '92*, 1992.
[10] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," *Proc. CCS '03*, Oct. 30, 2003.
[11] D. Liu, P. Ning and W. Du, "Group-Based Key Pre-Distribution in Wireless Sensor Networks," WiSE '05, Sept. 2005.
[12] A. Perrig *et al.*, "Efficient Authentication and Signing of Multicast Streams over Loosy Channels," *Proc. 2000 IEEE Symp. Sec. and Privacy*, 2000.
[13] K. Sun *et al.*, "TinySeRSync: Secure and Resilient Time Synchronization in Wireless Sensor Networks," *Proc. CCS '06*, Nov. 2006.
[14] B. Lai, S. Kim, and I. Verbauwhede, "Scalable Session Key Construction Protocol for Wireless Sensor Networks," *IEEE Wksp. Large Scale Real Time and Embedded Sys.*, 2002.
[15] S. A. Camtepe and B. Yener, "Key Distribution Mechanism for Wireless Sensor Networks: A Survey," Rensselaer Polytechnic Inst., TR-05-07, Mar. 2005.
[16] L. Buttyán and J-P Hubaux, "Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks," *ACM/Kluwer Mobile Networks and Apps.*, vol. 8, no. 5, Oct. 2003, pp. 291–301.
[17] N. Salem *et al.*, "A Charging and Rewarding Scheme for Packet Forwarding in Multi-hop Cellular Networks," *Proc. ACM MobiHoc*, 2003.

BIOGRAPHIES

JONGMIN JEONG (jj248@cornell.edu) received his B.Sc., M.Sc. and Ph.D. in computer information and communication engineering from Kangwon National University, Korea. He currently works as a post-doctoral fellow at Cornell University. His research interests include security of wireless networks such as cellular networks, ad hoc networks, sensor networks, WLANs, and RFID systems.

ZYGMUNT J. HAAS (haas@ece.cornell.edu) received his B.Sc. and M.Sc. in electrical engineering in 1979 and 1985, respectively. In 1988, after earning his Ph.D. from Stanford University, he joined AT&T Bell Laboratories, Network Research Department. There he pursued research on wireless communications, mobility management, fast protocols, optical networks, and optical switching. In August 1995 he joined the faculty of the School of Electrical and Computer Engineering at Cornell University, where he is now a professor and the associate director for Academic Affairs. He is an author of numerous technical conference and journal papers, and holds 18 patents in the areas of high-speed networking, wireless networks, and optical switching. He has organized several workshops, delivered numerous tutorials at major IEEE and ACM conferences, and has served as an editor of several journals and magazines, including *IEEE Transactions on Networking*, *IEEE Transactions on Wireless Communications*, *IEEE Communications Magazine*, and Springer's *Wireless Networks* journal. He has been a guest editor of *IEEE JSAC* issues on gigabit networks, mobile computing networks, and ad hoc networks. He served as Chair of the IEEE Technical Committee on Personal Communications and is currently serving as Chair of the Steering Committee of *IEEE Pervasive Computing*. His interests include mobile and wireless communication and networks, performance evaluation of large and complex systems, and biologically inspired networks. His URL is http://wnl.ece.cornell.edu.