Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2006, Article ID 90652, Pages 1–11
DOI 10.1155/WCN/2006/90652

SeGrid: A Secure Grid Framework for Sensor Networks

Xiuzhen Cheng,[1] Fang Liu,[1] and Fengguang An[2]

[1] *Department of Computer Science, The George Washington University, Washington, DC 20052, USA*
[2] *Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China*

Received 10 October 2005; Accepted 22 December 2005

In this paper, we propose SeGrid, a secure framework for establishing grid keys in low duty cycle sensor networks, for which establishing a common key for each pair of neighboring sensors is unnecessary since most sensors remain in sleep mode at any instant of time. SeGrid intends to compute a shared key for two grids that may be multihop away. This design explores the fact that for most applications, closer grids have higher probability and desire for secure message exchange. SeGrid relies on the availability of a low-cost public cryptosystem. The query and update of the corresponding public shares are controlled by a novel management protocol such that the closer the two grids, the shorter the distance to obtain each other's public share. We instantiate SeGrid based on Blom's key establishment to illustrate the computation of a grid key.

1. INTRODUCTION

Security provisioning is a critical service for many sensor network applications [1–3]. However, the severely constrained resources (memory, processor, battery, etc.) within a sensor render many of the popular public key-based security primitives inapplicable [4]. Therefore, much research effort [5–11] has been placed on how to establish a shared key between two sensors such that their communications can be secured with low-cost symmetric encryption techniques.

Most existing schemes [8] for key establishment in sensor networks intend to design light weight (in computational complexity) algorithms for computing pairwise keys between neighboring sensors. The induced key-sharing graph containing edges incident at two sensors sharing a common key should be globally connected in order for the network to function properly. Another constraint considered by these techniques is the memory budget allocated for a priori key information storage. The tradeoff between the consumed memory space versus the security of the scheme and the connectivity of the induced key-sharing graph has been well studied in many of these works.

As understood by the research society, the utmost problem in a sensor network is the operation time elongation. Even though the above-mentioned works do take resource (especially memory space) consumption into consideration, none of them explores the *density* dimension for further energy conservation. In this paper, we propose SeGrid, a grid-based framework for establishing grid keys in low duty cycle sensor networks. We envision that all sensors within a grid are equivalent in routing (as in [12]), and thus a secret key is needed between two grids (instead of two nodes) that demand secure communication. In SeGrid, only one or a few sensors (for fault tolerance) within a grid are active at any instant of time and all other sensors fall asleep for energy conservation. This design explores the fact that sensors are low cost and are densely deployed in a typical network. When a new sensor becomes active, or an active sensor dies due to energy depletion, the shared grid keys should be recomputed. Note that this is different than group key construction. If all sensors within a grid form a group, then SeGrid intends to compute a shared key between two group leaders, with the help of all active group members. We instantiate this idea by applying Blom's key establishment scheme [13] to demonstrate the grid key computation. Note that putting redundant sensors to sleep for energy conservation is a popular method in topology control [12, 14, 15] and energy-efficient protocol design [16–18]. However, to the best of our knowledge, this work is the first to combine energy-efficient topology control with key establishment.

This research is motivated by the following observations: two sensors that are closer have higher chance to exchange message; and it is unnecessary for each pair of sensors to establish a shared key in low duty cycle sensor networks [19, 20]. The basic idea of SeGrid is outlined as follows. We assume that there exists a public cryptosystem with low computation cost (e.g., Blom's key establishment scheme [13]) such that each sensor can be preloaded with a cryptopair containing a public share and a private share before deployment. In SeGrid, sensors compute the grids they are residing in and choose to sleep or wake up based on some schedule (e.g., the wake-up schedule proposed in [18]). Each grid has a *grid head*, an active sensor for message transmission and public share storage. The grid head stores the public shares of all active nodes within its grid at designated locations and queries the nearest grid that stores the public shares of another grid based on a novel public share management protocol. After obtaining the public shares of the destination grid, the source grid computes a key k_s that will be used to secure all transmissions between these two grids. The destination grid can follow the same procedure to compute the grid key k_s. The public share management protocol ensures that the closer the two grids are, the shorter the expected query distance to obtain each other's public shares is. This protocol involves only simple algebraic (shift and addition) operations, thus has very low computation overhead. We finally instantiate SeGrid based on Blom's key establishment scheme [13] to demonstrate how the grid key can be computed based on the underlying public cryptosystem.

The features of SeGrid and the contributions of this paper are summarized as follows.

(i) SeGrid divides sensors into a grid structure and realizes a secure grid communication with only a few number of nodes being active in each grid. The majority of the sensors fall asleep for energy conservation, and rely on the associated grid heads for intergrid secure communication. This design is extremely useful for energy constrained sensor networks. To our best knowledge, SeGrid is the first work that considers key establishment and topology control for energy conservation at the same time.

(ii) SeGrid can be easily applied to multihop end-to-end secure communication. Existing key establishment schemes rely on intermediary sensors for path key computation to construct a shared key between two sensors that are multihop away. Path keys are vulnerable because they are exposed to all intermediary nodes, violating the security requirement that a pairwise secret should be known to only the communicating pairs.

(iii) The required storage space per grid is proportional to $\log \sqrt{N}$, where N is the total number of grids in the network. This indicates that memory space consumed by SeGrid grows very slowly with the increase of the network size.

(iv) The proposed grid-based public share management scheme explores the communication overhead tradeoff between queries and updates for public share management in SeGrid. This design investigates the fact that in many sensor network applications, two grids that are farther away have weaker desire to communicate directly.

This paper is organized as follows. Section 2 briefly outlines the most related works. Network model and the underlying assumptions are elaborated in Section 3. We propose our grid-based framework for establishing grid keys in sensor networks (SeGrid) in Section 4. The performance of SeGrid is studied in Section 5. An example instantiation of SeGrid is sketched in Section 6. We conclude this paper with a discussion in Section 7.

2. RELATED WORK

Since the pioneer work of Eschenauer and Gligor [9], many researchers have been working on how to bootstrap shared keys for two sensors that desire secure communication. In this section, we summarize the major related works along the lines of *random key/keying information predistribution* and *in situ pairwise key establishment*.

The basic random key predistribution scheme is proposed by Eschenauer and Gligor in [9], in which a large key pool $\mathcal{K}$ is computed offline and each sensor picks k keys randomly from $\mathcal{K}$ without replacement to form a key ring before deployment. Two sensors can establish secure communication as long as they have at least one common key in their key rings. To enhance the security of the basic scheme, Chan et al. [6] propose the q-composite keys scheme in which $q > 1$ number of common keys are required for two nodes to establish a shared key. To improve scalability, Du et al. [8] employ the group deployment concept, in which sensors are grouped before deployment and each group is dropped at one deployment point. Correspondingly the large key pool $\mathcal{K}$ is divided into subkey spaces, with each associated with one group of sensors. Subkey spaces overlap if the corresponding deployment points are adjacent. Such a scheme ensures that close-by sensors have higher chance to establish a pairwise key directly. In all these schemes [6, 8, 9], a path key can be established for two neighboring sensors that demand secure communication but have no common keys in their key rings. A drawback of this mechanism is that the path key is exposed to all intermediary nodes. To overcome this problem, Zhu et al. [11] propose to break the secret (the shared key) into multiple shares and each share is delivered to the destination along a different logical path. The secret is restored at the destination when a number of shares are received.

Note that none of the above mentioned random key predistribution schemes guarantees that a key is shared by only one pair of sensors. Therefore compromising one sensor may threaten links that are incident to uncompromised nodes. This problem has been tackled by Chan et al. in [6, 21], which propose the random pairwise keys scheme. In this scheme, every node receives a number of unique keys, with each uniquely shared with another node that is randomly selected [6] or selected based on a virtual grid [21] before deployment. This pairing is done based on node IDs, and therefore mutual authentication can be realized after deployment since all keys are unique and each is associated with a pair of nodes. A path key can be established with the help of one or more trusted intermediaries [21]. Combining the concepts of random pairwise keys and group deployment, the two independently proposed but similar key establishment schemes by Liu et al. [22] and by Zhou et al. [23] have better scalability and lower storage overhead.

To further improve security and scalability, a couple of random key space predistribution schemes [7, 10] have been proposed. These two schemes are very similar in nature, except that the key spaces are defined differently. In [7], a key space is constructed based on Blom's method [13], and a shared key between two nodes corresponds to one entry of a symmetric matrix. In [10], a key space is defined by a symmetric bivariate t-degree polynomial [24], and the shared key of two sensors is the value obtained by plugging the two IDs into a polynomial. In both schemes, a number of key spaces are precomputed and each sensor is associated with one or more key spaces before deployment. Two sensors can compute a pairwise key after deployment if they have keying information from a common key space.

As claimed by [25], random key and key space predistribution schemes explore the tradeoff of security and memory consumption, since the amount of preloaded information is constrained by the memory budget within each sensor. A stronger security results in higher memory consumption. This seems unavoidable in all predistribution schemes, due to the randomness since no sensor network topology information is available before deployment.

iPAK [26] and SBK [27], two truely in situ key establishment schemes that remove the randomness, achieve good security with a small amount of memory consumption. In iPAK and SBK, a number of service sensors are sacrifices and therefore worker sensors do not need any predeployment knowledge for pairwise key establishment. In iPAK, service sensors, with each carrying a key space, and worker sensors, with no a priori knowledge, are deployed at the same time. In SBK, homogeneous sensors are preloaded with several system parameters and they differentiate their roles as either service nodes or worker nodes after deployment. A key space is constructed after the role of a service node is determined. In both schemes, worker sensors obtain security information through an asymmetric secure channel from service nodes and then compute shared keys with their neighbors. Each service node has a λ-secure key space, and distributes keying information to at most λ worker sensors through an asymmetric secure channel established by Rabin's algorithm [28]. Compared to iPAK, SBK is "perfect" against node capture attack, achieves high connectivity (close to 1) in the induced key-sharing graph, and consumes a small amount of memory in worker sensors.

SeGrid is different from all those mentioned above in that secure communication is realized based on the shared keys between two grids instead of two sensors. SeGrid divides the sensor network into a virtual grid structure based on location information, and computes a location-aware grid key between any two grids. Only one or a few number of sensors are active at any instant of time in each grid, with one of them as the grid head. All the intergrid communications must be directed through the associated grid heads. SeGrid is able to provide multihop end-to-end secure communication, and thus does not require the establishment of a path key. SeGrid considers topology control for energy conservation and key establishment at the same time, a practical solution for network lifetime elongation. In SeGrid, memory consumption grows very slowly when the size of the network increases fast (proportional to $\log \sqrt{N}$); therefore, SeGrid has good scalability.

3. NETWORK MODEL AND ASSUMPTIONS

We consider a sensor network deployed in outdoor environments. Each sensor is able to position itself through any of the techniques proposed in literature (e.g., [29–31]). A virtual grid will be computed based on position information and each sensor resides in one grid. The ID of a grid is denoted by (X, Y). At any instant of time, one or $t > 1$ number of sensors, where t is a small integer, are active within a grid and all other sensors fall asleep for energy conservation. A sleeping sensor wakes up periodically in order to replace a sensor with depleted energy. An active sensor is in full operation and all active sensors collaborate together to guarantee the functioning of the network. Sensors within neighboring grids can communicate directly. The wake-up/sleep schedule, the active/inactive status transition, and the underlying routing protocol for message dissemination are out of the scope of this paper. We just simply assume that they are available for us to employ. Existing works that are related to these topics can be found in [18, 32], and so forth.

We will explore a public cryptosystem that contains public and private crypto-pairs. The public share in a crypto-pair can be disseminated to the public as plain text while its corresponding private share must be kept secret. By exchanging their public shares, two nodes can compute a shared key based on their private shares and the exchanged public shares. Examples of cryptosystems satisfying these conditions include the Diffie-Hellman key exchange protocol [33], the symmetric matrix-based key establishment scheme [13], and the polynomial-based scheme [24]. In Section 6, we are going to instantiate SeGrid based on Blom's method.

We assume each sensor is preloaded with a crypto-pair before deployment. The operation of the sensor network is unattended after deployment. Each grid may have more than one public share, if it has more than one active sensor. An update message will be directed to all locations storing the public shares of the grid such that the public shares of newly introduced active (old inactive) sensors can be inserted (removed). A grid demanding the public shares of another grid can just query the nearest grid storing the corresponding information. We will propose a simple protocol for public share management in Section 4.2.

We envision that in a sensor network all nodes within a grid are equivalent. Therefore we only consider the secure communication between two grids. The computation of the shared key k_s between the two grids depends on the underlying public cryptosystem. We will show how to compute k_s based on Blom's key establishment scheme in Section 6. Note that intragrid secure communications are needed when more than one sensor is active simultaneously within a grid. The shared keys between these active nodes can be computed based on the underlying public cryptosystem too.

Note that even though $t > 1$ number of sensors may be active at any instant of time, we assume that only one sensor within a grid is in charge of all intergrid communications and public share storage. This sensor is the *grid head*. In other words, the grid head stores all public shares for the associated grid. Note that a grid head will be replaced by a new one when its energy is depleted. This procedure is out of the scope of this paper too. Existing works that cover the role transition of grid heads can be found in [12, 16]. The new grid inherits all stored public shares from the old one for the associated grid.

4. SEGRID: THE GRID-BASED FRAMEWORK FOR KEY ESTABLISHMENT

In this section, we propose the basic idea of SeGrid, a grid-based framework for key establishment in sensor networks. Note that this elaboration does not depend on any public cryptosystem. We will instantiate this idea in Section 6 based on Blom's key establishment scheme [13].

In SeGrid, each sensor computes the associated grid ID locally and independently based on its position information according to the *grid and grid head determination* scheme. Therefore all sensors are partitioned based on a virtual grid structure after deployment. All active nodes within a grid store their public shares at designated locations (grids) determined by the *public share management* scheme. When two grids need to set up their shared key, each grid first figures out the nearby location from which to query the public shares of the other grid, and then applies the *grid key computation* technique. After a secret is computed, two grids can securely communicate with each other to protect all message exchange.

SeGrid considers both key establishment and network lifetime extension through topology control simultaneously. With only one or a few number of active nodes in each grid, the majority of the sensors can sleep most of the time and rely on the associated grid heads for grid-to-grid communication. The novel public share management scheme ensures that two grids get the public shares of each other from a position within a short distance. A shared key between two grids can be further secured with the location information.

In the following, we will first describe a simple algorithm for each sensor to locally and independently compute the ID of the grid in which it resides. Then we give a novel protocol for each grid to determine where to store its own public shares, and also where to obtain the public shares of the other communicating grid to establish a shared grid key. In the last, we propose how to apply SeGrid for protecting the unicast communications between two grids.

4.1. Grid and grid head determination

In GAF [12], the size of a grid is determined based on node equivalence for routing. In other words, any node within a grid can communicate directly with any other node in any neighboring grid. This constraint specifies that the size of a grid, denoted by r, can be at most $R/\sqrt{5}$, where R is the nominal transmission range. In our study, we adopt this idea since we also intend to turn off most of the sensors within a grid for energy conservation in order to extend network lifetime. GAF specifies the length of the grid edge but does not specify how to determine the grid a node resides in. In the following, we propose a very simple algorithm to allow each node independently and locally determine its grid.

Assume a sensor S is deployed at position (x, y). Then the grid ID (X, Y) where S resides in can be derived as

$$\begin{aligned} X &= \lfloor x \div 2^{\lfloor \log_2 r \rfloor} \rfloor, \\ Y &= \lfloor y \div 2^{\lfloor \log_2 r \rfloor} \rfloor. \end{aligned} \tag{1}$$

Note that the grid ID (X, Y) can be computed through shift operations only, as long as $\lfloor \log_2 r \rfloor$ is computed offline and uploaded into each sensor before deployment. This is a reasonable assumption since r depends only on the nominal transmission range R, which can be made available before deployment. Therefore we can simply shift the binary representations of x and y to the right for k positions, where $k = \lfloor \log_2 r \rfloor$, to obtain X and Y.

If only one active sensor is required within a grid, the protocol proposed in [12] for active node selection suffices. In this case, the unique active sensor serves as the grid head. When $t > 1$ active sensors per grid are required, these nodes can be elected based on node ID, or residual power. For example, a simple protocol may require that the t sensors with the smallest t IDs in a grid whose residual powers are above some threshold remain active while others go to sleep after the grid ID of each sensor is computed and broadcasted. In this case, we can choose the sensor with minimum ID as the grid head. Requiring more than one active sensor per grid provides better fault tolerance since the grid head is in charge of both message dissemination and keying information storage. When a grid head needs to be replaced due to reasons such as power depletion, it can delegate another active sensor as the new grid head and transfer all stored public shares before turning to sleep mode. If no active sensor within the same grid is available, the grid head should wait until a sleeping sensor wakes up. A similar procedure for grid head role transition has been proposed in [12].

4.2. Public share management

As stated before, each grid needs to store the public shares of all its active nodes at designated positions (grids) for the convenience of being queried by other grids to establish intergrid shared keys. To solve this problem, we need to answer two questions. First, for any grid (X_0, Y_0), where shall we store its public shares? Second, if grid (X_1, Y_1) would like to securely communicate with (X_0, Y_0), where to find out the latter's public shares? We propose a simple protocol for storing and querying the public shares of a grid.

Our protocol is based on the following assumption: the closer the two grids are, the higher the probability they may communicate is. Therefore, the public shares of a grid will be stored at designated locations such that the closer the location is to the grid, the shorter the expected query distance involved in public share acquisition is. In our protocol, the density of the grids storing the public shares of a grid drops logarithmically as the distance to the grid increases. Figure 1

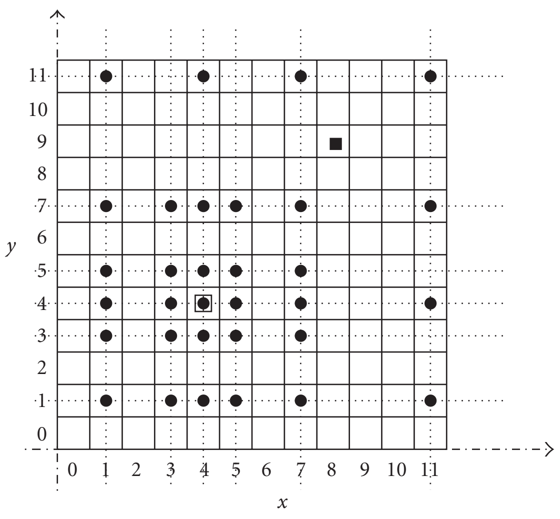

Figure 1: The public shares of the grid (4,4) will be stored at (4,4), (3,4), (4,3), (4,5), (5,4), (3,3), (5,5), (3,5), (5,3), (1,1), (3,1), (4,1), (5,1), (7,1), (1,3), (7,3), (1,4), (7,4), (1,5), (7,5), (1,7), (7,7), (3,7), (4,7), (5,7), (1,11), (4,11), (7,11), (11,11), (11,7), (11,4), (11,1). If the grid (8,9) needs the public shares of (4,4), it can query either (7,7) or (7,11) since they are closer.

gives a simple example to illustrate the storage locations of the public shares for the grid (4,4).

The answer to the first question is very simple. The public shares of the grid (X_0, Y_0) will be stored at the grid (X, Y) where

$$\begin{gathered} x = X_0, \\ Y = Y_0 \pm (2^{i+1} - 1), \quad \text{for } i = 0, 1, \ldots, \end{gathered} \tag{2}$$

or

$$\begin{gathered} X = X_0 \pm (2^{i+1} - 1), \quad \text{for } i = 0, 1, \ldots, \\ Y = Y_0 \end{gathered} \tag{3}$$

or

$$\begin{aligned} X &= \begin{cases} X_0 \pm (2^i - 1), \\ X_0 \pm (2^{i+1} - 1), \end{cases} \quad \text{for } i = 0, 1, \ldots, \\ Y &= \begin{cases} Y_0 \pm (2^i - 1), \\ Y_0 \pm (2^{i+1} - 1), \end{cases} \quad \text{for } i = 0, 1, \ldots. \end{aligned} \tag{4}$$

To identify the nearest grid that stores the public shares of (X_0, Y_0), the grid (X_1, Y_1) computes

$$\begin{aligned} i_x^- &= \lfloor \log_2 (|X_1 - X_0| + 1) \rfloor, \\ i_x^+ &= \lceil \log_2 (|X_1 - X_0| + 1) \rceil, \\ i_y^- &= \lfloor \log_2 (|Y_1 - Y_0| + 1) \rfloor, \\ i_y^+ &= \lceil \log_2 (|Y_1 - Y_0| + 1) \rceil. \end{aligned} \tag{5}$$

Note that the grids formed by

$$\begin{aligned} X &= X_0 + \operatorname{sign}(X_1 - X_0) \times (2^{i_x} - 1), \\ Y &= Y_0 + \operatorname{sign}(Y_1 - Y_0) \times (2^{i_y} - 1), \end{aligned} \tag{6}$$

where $i_x = i_x^-$ or i_x^+, $i_y = i_y^-$ or i_y^+,

$$\operatorname{sign}(x) = \begin{cases} 1 & \text{if } x \ge 0, \\ -1 & \text{if } x < 0, \end{cases} \tag{7}$$

and $|i_x - i_y| \le 1$ if $i_x \ne 0$ and $i_y \ne 0$, store the public shares of (X_0, Y_0). Therefore (X_1, Y_1) can choose the nearest one to query the public shares of (X_0, Y_0). An example is given in Figure 1 when grid (8,9) queries the public shares of (4,4). Based on (5), $i_x^- = 2$, $i_x^+ = 3$, $i_y^- = 2$, and $i_y^+ = 3$. The nearest grids storing (4,4)'s public shares are either (7,7) or (7,11) since they are the closest among the 4 grids formed by $X = 4 + 3$, $X = 4 + 7$, $Y = 4 + 3$, and $Y = 4 + 7$.

As shown by (2)–(4), the computation of the storage locations for a grid contains only shift and addition operations. However, the identification of the nearest grid for public share query (see (5)) involves the complicated log functions. Nevertheless, this can be done easily through a lookup table. Hence, each grid can easily determine where to store and query the public shares.

Note that if Manhattan distance instead of Euclidean distance is used as a routing metric for public share queries and updates, the computation overhead is further decreased since only simple addition and subtraction operations are involved.

Remarks 1. (i) This protocol guarantees that closer grids obtain the public shares within shorter distance. Therefore, the farther away the two grids are, the higher the communication overhead for their public share queries is. In reality, closer grids intend to communicate with higher probability.

(ii) The update of the public shares for a grid always takes the same number of messages, as long as the routing protocol remains unchanged.

(iii) The grid head will store all public shares for other grids based on (2)–(4). Before turning to sleep mode, the grid head should transfer all stored information to the new grid head.

4.3. Grid key computation

In SeGrid, two communicating grids need to establish a shared key computed by obtaining the public shares of each other. The computation of the grid key can be further secured with the grid location information. However, the detailed computation procedure depends on the underlying public cryptosystem. In Section 6, we will show how to compute the shared key between two grids based on Blom's key establishment scheme [13].

4.4. *Secure grid communication*

Now we are ready to propose our secure grid communication scheme. We assume there exists a routing protocol, either geography-based (e.g., [34]) or topology-based (e.g., [32]), such that we can employ directly.

Recall that SeGrid is built upon a public cryptosystem that contains public and private crypto-pairs. By exchanging their public shares, two sensors can establish a shared secret based on the private shares and the exchanged public shares. Therefore two nodes within the same grid can communicate securely with each other. However, intergrid secure communications must seek the help of the grid heads, as illustrated by the following procedure.

(1) Each active sensor first establishes a secure intragrid communication link with the associated grid head. The nodes exchange their public shares, and compute the shared key with their private shares.

(2) The two corresponding grid heads are responsible for the secure intergrid communication. Each grid head first queries the nearest grid that contains the public shares of the other party based on the procedure proposed in Section 4.2 to obtain the public shares, then computes a secret key k_s shared by these two grids. k_s will be used to secure all the future communications between the two grids.

5. PERFORMANCE ANALYSIS

In this section, we study the performance of SeGrid in terms of memory overhead, communication cost, and resilience against node capture attack. Note that in SeGrid, only a few number of sensors are active in each grid at any instant of time, and are involved in grid key computation. Let S denote the sensor network under consideration and let N be the total number of grids in S. For simplicity we assume that all grids form a square region, and each grid has an edge length of “1” unit.

5.1. *Storage overhead*

In the proposed SeGrid framework, the public shares of each grid need to be stored at designated grids for the convenience of being queried by other grids to establish shared grid keys. In this subsection, we study the storage overhead, that is, the maximum number of public shares a grid head may store for other grids. To simplify the analysis, we assume each grid has only one active sensor, the grid head. Therefore each grid stores at most one public share for another grid. Let τ be the maximum number of public shares a grid stores in a sensor network S.

Lemma 1. *When $N = 2^{2k} - 2^{k+1} + 1$, where $k = 1, 2, \ldots$, the grid in the center stores τ public shares in the network. Further, $\tau = 1$ for $k = 1$, and $\tau = 16k - 23$ for $k > 1$.*

Proof (Induction). When $k = 1$, only one grid exists. It is obvious that the lemma holds true. When $k = 2$, $N = 9$,

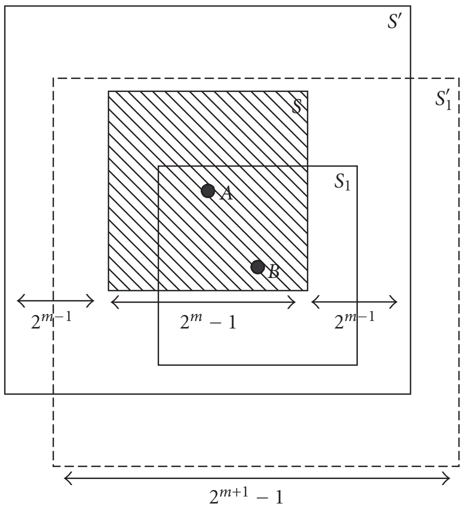

Figure 2: A is the central grid and stores the maximum number of public shares in S. When S is enlarged into S' and N is increased from $2^{2m} - 2^{m+1} + 1$ to $2^{2(m+1)} - 2^{m+2} + 1$, A's public shares stored in S' are still the maximum.

the center grid stores a copy for each of the other 8 grids, while a boundary grid stores less based on the public share management protocol. Therefore the lemma holds true since $16 \times 2 - 23 = 9$.

Assume the argument holds true until $k = m$. Now consider $k = m+1$. The network is enlarged from S with an edge length of $2^m - 1$ to S' with an edge length of $2^{m+1} - 1$. Let A be the central grid of both S and S', then A stores the maximum number of public shares in S. For contradiction, we assume that another grid B other than A stores the maximum number of public shares in S'.

For grid A, the public shares stored in S' come from two sources: the public shares A stores in the $16m - 23$ grids of S from the assumption, and the public shares in the newly enlarged area. According to (2)–(4), 16 public shares are added to the area defined by $S' - S$, (2)–(4) show that 16 more grids are included whenever m is increased by 1 for all $m > 1$. Thus, A stores $16(m + 1) - 23$ copies of public shares in S'.

The public shares stored at B come from two sources too. As indicated by Figure 2, B stores at most $16m - 23$ number of public shares within the area S_1 whose edge length is $2^m - 1$, since S_1 overlap with S'. B also stores public shares from the area $S'_1 - S_1$, where S'_1 is centered at B. According to (2)–(4), this area contains less than 16 public shares of B. Therefore the total number of B's public shares in S' is less than $(16m - 23) + 16$, that is, $16(m + 1) - 23$. This contradicts with the previous assumption. Thus, B cannot store the maximum number of public shares in the enlarged network S'. □

Corollary 1. *When $2^{2(k-1)} - 2^k + 1 < N < 2^{2k} - 2^{k+1} + 1$, where $k = 2, 3, \ldots$, then the maximum number of public shares stored by a grid in the network is at most $16k - 23$.*

Proof. This corollary holds true from Lemma 1 and (2)–(4).

Based on Lemma 1 and Corollary 1, we obtain the following theorem. □

Theorem 1. *Let N be the total number of grids in a network following the proposed public share management scheme, where $N > 9$. Then the number of public shares stored at each grid is at most $16 \times \lceil \log_2(\sqrt{N}+1) \rceil - 23$.*

Proof. For any $N > 9$, there exists an integer k satisfying $k > 1$ such that

$$2^{2(k-1)} - 2^k + 1 < N \le 2^{2k} - 2^{k+1} + 1,$$
$$\text{that is, } 2^{k-1} - 1 < \sqrt{N} \le 2^k - 1, \tag{8}$$
$$\text{that is, } k - 1 < \log_2(\sqrt{N}+1) \le k.$$

Therefore $k = \lceil \log_2(\sqrt{N}+1) \rceil$. According to Lemma 1 and Corollary 1, each grid stores at most $16k - 23$ public shares in the network, which means that the maximum storage per grid is at most $16 \times \lceil \log_2(\sqrt{N}+1) \rceil - 23$. □

Theorem 2. *For a network following the public share management scheme, the number of public shares stored at a grid A is equal to the total number of A's public share stored within the network.*

Proof. Let (X, Y) be a grid that stores the public share of the grid (X_0, Y_0) in the network. Assume $X \ne X_0$ and/or $Y \ne Y_0$. Rewrite (4) as follows:

$$X = \begin{cases} X_0 \pm (2^i - 1), \\ X_0 \pm (2^{i+1} - 1), \end{cases}$$
$$Y = \begin{cases} Y_0 \pm (2^i - 1), \\ Y_0 \pm (2^{i+1} - 1), \end{cases} \tag{9}$$

where $i = 0, 1, 2, \ldots$. It follows that

$$X_0 = \begin{cases} X \mp (2^i - 1), \\ X \mp (2^{i+1} - 1), \end{cases}$$
$$Y_0 = \begin{cases} Y \mp (2^i - 1), \\ Y \mp (2^{i+1} - 1), \end{cases} \tag{10}$$

Hence for any grid B, B stores A's public share if and only if A stores B's public share. From (2) and (3), it is easy to argue that the same relationship holds true for the cases of $X = X_0$ and/or $Y = Y_0$. □

According to Theorems 1 and 2, the storage overhead required in each grid is at most $16 \times \lceil \log_2(\sqrt{N}+1) \rceil - 23$, where $N > 9$. This indicates that in the worst case, storage overhead increases very slowly when N grows fast. Figure 3 plots the average number of public shares stored at each grid obtained from simulation study as well as the previously computed theoretical upper bound. Both trends imply that the storage in each grid grows slowly as the number of grids increases.

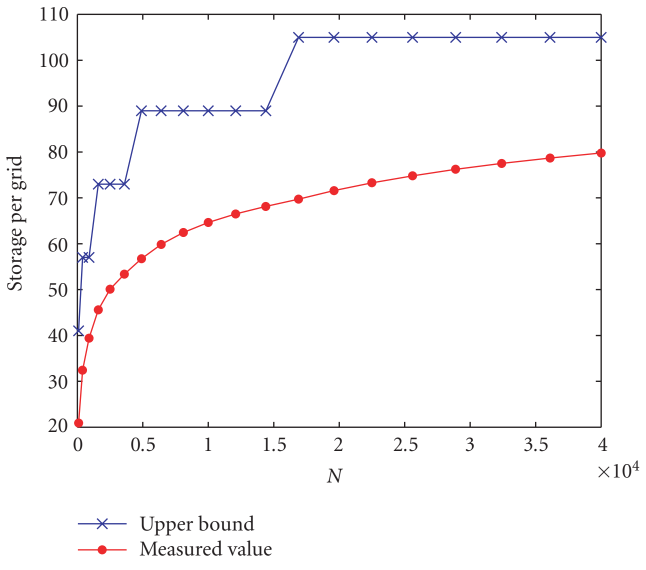

FIGURE 3: The memory storage for public shares at each grid.

5.2. Communication overhead

In SeGrid, public shares of each grid need to be stored at designated locations at the system initialization phase and to be updated later when sensors change state. Further, querying public shares of another grid also involves message transmission.

Storing public shares of each grid during the initialization phase contributes the most to the communication overhead, since each grid needs to store a copy of its public shares in every designated position according to the public share management scheme. Nevertheless, public shares can be transmitted just in plain texts, and can be very small (e.g., as implemented by [7], only the seed of the public share needs to be exchanged). On the other hand, SeGrid explores the communication overhead tradeoff between public share queries and updates. An update happens only when there is an active membership change, and this update may travel long distance. However, query overhead can be decreased since no global flooding will be involved. For a system with frequent public share acquisition request, the proposed public share management protocol is efficient in energy conservation.

5.3. Resilience against node capture

SeGrid relies on the availability of the underlying public cryptosystem for shared key computation between two sensors in the network. By compromising a number of sensors, an attacker may obtain the grid key and conduct further attacks. The security of SeGrid is dependent on the underlying public cryptosystem.

For example, the security of the grid key computation method proposed in Section 6.2 is constrained by the

λ-security of the Blom's key establishment system. Once more than λ number of sensors are compromised, the whole system becomes insecure. Increasing λ does improve security, but this requires a larger amount of memory. A possible strategy to overcome this problem is to hierarchically apply multiple key spaces. We target this as a future research.

6. A SIMPLE REALIZATION

In this section, we provide a simple realization of SeGrid for sensor networks based on Blom's key establishment scheme [13]. For completeness, we give a brief overview on Blom's scheme first. Then we describe how to compute a grid key based on Blom's scheme. Finally, we propose a location-aware grid key computation as an enhancement.

6.1. Preliminary: Blom's key management scheme

Blom's λ-secure key establishment scheme [13] has been well tailored for light-weight sensor networks by [7]. In the following, we will give an overview on Blom's scheme based on [7].

Let G be a $(\lambda + 1) \times M$ matrix over a finite field $GF(q)$, where q is a large prime. The connotation of M will become clear later. G is public, with each column called a public share. Let D be any random $(\lambda + 1) \times (\lambda + 1)$ symmetric matrix. D must be kept private, which is known to the network service provider only. The transpose of $D \cdot G$ is denoted by A. That is, $A = (D \cdot G)^T$. A is private too, with each row called a private share. Since D is symmetric, $A \cdot G$ is symmetric too. If we let $K = (k_{ij}) = A \cdot G$, we have $k_{ij} = k_{ji}$, where k_{ij} is the element at the ith row and the jth column of matrix K, $i, j = 1, 2, \ldots, M$.

The basic idea of Blom's scheme is to use k_{ij} as the secret key shared by node i and node j. D and G jointly define a *key space* (D, G). Any public share in G has a unique private share in A, which form the so-called *crypto-pair*. For example, the ith column of G, and the ith row of A form a crypto-pair and the unique private share of the ith column of G, a public share, is the ith row of A. Two sensors whose crypto-pairs are obtained from the same key space can compute a shared key after exchanging their public shares. From this analysis, it is clear that M is the number of sensors that can compute their pairwise keys based on the same key space.

In summary, Blom's scheme states the following protocol for nodes i and j to compute k_{ij} and k_{ji}, based on the same key space.

(i) Each node stores a unique crypto-pair. Without loss of generality, we assume node i gets the ith column of G and the ith row of A, denoted by g_{ki} and a_{ik}, where $k = 1, 2, \ldots, \lambda + 1$, respectively. Similarly, node j gets the jth column of G and the jth row of A, denoted by g_{kj} and a_{jk}, where $k = 1, 2, \ldots, \lambda + 1$, respectively.

(ii) Node i and node j exchange their stored public shares drawn from their crypto-pairs as plain texts.

(iii) Node i computes k_{ij} as follows:

$$k_{ij} = \sum_{k=1}^{\lambda+1} a_{ik} \cdot g_{kj}. \tag{11}$$

Similarly, node j computes k_{ji} by

$$k_{ji} = \sum_{k=1}^{\lambda+1} a_{jk} \cdot g_{ki}. \tag{12}$$

Blom's key establishment scheme ensures the so-called λ-secure property, which means that the network should be perfectly secure as long as no more than λ nodes are compromised. This requires that any $\lambda + 1$ columns of G must be linearly independent. An interesting method of computing G is proposed by Du et al. in [7]. This idea is sketched as the following. Let *len* be the number of bits in the symmetric key to be computed. Choose q as the smallest prime that is larger than 2^{len}. Let s be a primitive element of $GF(q)$ and $M < q$. Then

$$G = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ s & s^2 & s^3 & \cdots & s^M \\ s^2 & (s^2)^2 & (s^3)^2 & \cdots & (s^M)^2 \\ & & & \vdots & \\ s^\lambda & (s^2)^\lambda & (s^3)^\lambda & \ldots & (s^M)^\lambda \end{bmatrix} \tag{13}$$

Note that G is a Vandermonde matrix. Each column of G represents the public share of some sensor node storing that column. In Blom's key establishment scheme, public shares need to be exchanged between sensors that require secure peer-to-peer communication. Based on the structure of G, we observe that only the second element of each column, the *seed* of the column, needs to be stored and exchanged. Thus both storage and communication overheads can be greatly decreased.

6.2. Grid key computation based on Blom's method

Assume that a large key space (D, G) following Blom's key establishment scheme has been computed offline. Before deployment, each sensor receives a crypto-pair from the key space. Note that we do not require the crypto-pairs to different sensors to be unique, but we require that all active sensors within one grid have different crypto-pairs. It is possible that more than one active node exists in each grid, thus the key shared by two grids may be computed based on multiple public shares.

Let $t_A(t_B)$ be the number of active sensors in a grid $(X_A, Y_A)((X_B, Y_B))$. Following the public share management protocol proposed in Section 4.2, all these $t_A(t_B)$ public shares will be stored at designated grids, and are available to other grids upon a request. If (X_A, Y_A) and (X_B, Y_B) need secure communication, the grid head of (X_A, Y_A) computes and queries the nearest grid holding (X_B, Y_B)'s public shares, and distributes them to all the active nodes in the grid

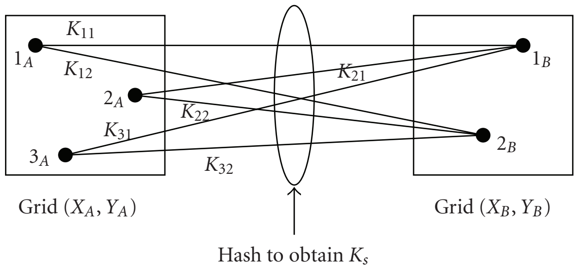

FIGURE 4: There exist three active sensors in grid (X_A, Y_A) and two active sensors in grid (X_B, Y_B). The two nodes labeled by 1_A and 1_B are the grid heads in the corresponding grids. After obtaining the public shares of (X_B, Y_B), nodes 1_A, 2_A, and 3_A in grid (X_A, Y_A) compute k_{11} and k_{12}, k_{21} and k_{22}, and k_{31} and k_{32}, the shared keys with the two nodes in grid (X_B, Y_B) independently. Then each node i_A within (X_A, Y_A) computes $k_{i_A} = \text{Hash}(k_{i1}, k_{i2})$ for $i = 1, 2, 3$. This value will be securely transmitted to node 1_A. After obtaining all k_{i_A}'s, where $i = 1, 2, 3$, node 1_A computes k_s as $k_s = \text{Hash}(k_{1_A}, k_{2_A}, k_{3_A})$. Similarly, node 1_B in grid (X_B, Y_B) computes k_s based on the public shares of (X_A, Y_A).

(X_A, Y_A). The grid head of (X_B, Y_B) conducts the same procedure.

Now the grid key shared by (X_A, Y_A) and (X_B, Y_B) is ready to be computed independently in each grid based on the exchanged public shares. In Blom's key establishment scheme, two sensors can compute a shared key as long as they know each other's public share. We can derive a shared key k_s between two grids from the keys shared by all pairs of sensors within the two grids, as shown in Figure 4.

Let us use grid (X_A, Y_A) as an example to demonstrate the procedure of computing a shared key with the grid (X_B, Y_B). After obtaining the public shares of grid (X_B, Y_B) (consisted of the public shares from nodes 1_B and 2_B), each node i in (X_A, Y_A) computes a shared key with each node j in grid (X_B, Y_B). These pairwise keys are denoted by k_{ij}, where $i = 1, 2, \dots, t_A$ and $j = 1, 2, \dots, t_B$. Then each node i computes $k_i = \text{Hash}(k_{i1}, \dots, k_{it_B})$. This value is securely transmitted to the grid head h of (X_A, Y_A) based on the shared key between nodes i and h. After receiving all k_i's, where $i = 1, 2, \dots, t_A$, h derives the grid key k_s by computing $\text{Hash}(k_1, k_2, \dots, k_{t_A})$. The same procedure is conducted at the grid (X_B, Y_B). Note that the hash function exploited must be linear, and must be able to take arbitrary number of inputs. The simple XOR function is an example.

All affected grid keys must be reestablished whenever a new sensor becomes active or an old sensor dies due to energy depletion. Note that only the public shares of the node with role change needs to be updated (inserted or removed from designated grids).

Remarks 2. (i) The private shares of each sensor must be kept secret.

(ii) The security of the grid key computation protocol based on Blom's key establishment scheme [13] is determined by the λ-secure property of the key space (D, G). Therefore if the crypto-pairs of more than λ sensors are exposed to the adversary, the security of the whole network is compromised. This is the major drawback of applying Blom's key establishment scheme for grid key computation since the memory budget within a sensor for security information storage is limited.

(iii) The space consumed for storing the crypto-pairs within a sensor is related to λ. The larger the λ is, the higher the security level is, and the larger the storage space is.

(iv) The computation overhead of a grid key is determined by λ too. Each shared key computation between two active nodes takes $\lambda + 1$ number of modular multiplications.

6.3. Location-aware grid key enhancement

For the purpose of secure grid communication, the grid keys are desired to be unique. However, sensors may receive the same crypto-pairs in our realization based on Blom's method. Therefore two pairs of grids may have the same grid key.

Let G_1 and G_2 be two grids that compute a secure grid key $k_s(G_1, G_2)$. Assume there are altogether $n(\geq 2)$ active nodes in these two grids. Let $c_1, \dots, c_n$ denote the crypto-pairs associated with these n active nodes. Let G_1' and G_2' be another pair of grids containing n active nodes. The grid key $k_s(G_1', G_2')$ is to be computed based on the associated crypto-pairs $c_1', \dots, c_n'$. With the Blom's grid key computation scheme, the probability that the two pairs of grids derive the same grid key can be estimated as

$$\begin{aligned}
&\Pr(k_s(G_1, G_2) = k_s(G_1', G_2')) \\
&\quad \leq \Pr\left(\bigcup \{c_1, \dots, c_n\} = \bigcup \{c_1', \dots, c_n'\}\right) \\
&\quad = \Pr\left(c_1 = c_1', \dots, c_n = c_n'\right) \times n! \\
&\quad = \frac{n!}{M^n},
\end{aligned} \tag{14}$$

where M is the total number of crypto-pairs in the key space.

Figure 5 plots the probability that two pairs of grids compute the same grid key. We observe that a larger M or a larger n results in a lower probability. However, a larger n may shorten the network lifetime. In the following, we propose to apply grid position information for unique grid key derivation.

Assume that grid G_1 wants to compute its shared key with grid G_2. After G_1's grid head h has collected the confidential contributions $k_1, k_2, \dots, k_t$ from all the active nodes within G_1, h computes the grid key as

$$\text{Hash}(k_1, k_2, \dots, k_t, X_1, Y_1, X_2, Y_2), \tag{15}$$

where $(X_1, Y_1)((X_2, Y_2))$ is the grid ID of G_1 (G_2) computed from (1).

This position-aware grid key computation eliminates the ambiguity existing in the original grid key computation scheme based on Blom's method. By applying the unique ID of each grid, every pair of two grids can compute a unique shared key.

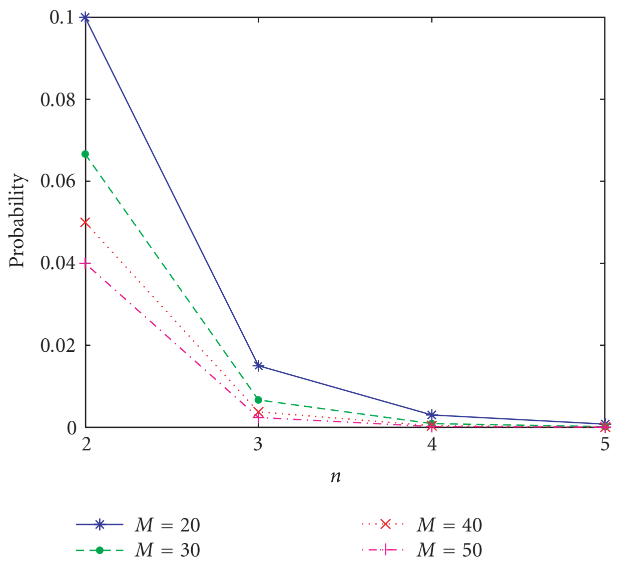

FIGURE 5: The probability that two pairs of grids obtain the same grid key.

7. CONCLUSION AND FUTURE RESEARCH

In this paper, we have proposed SeGrid, a grid-based key establishment framework for sensor networks. We have instantiated SeGrid based on Blom's key establishment scheme to demonstrate how to compute a grid key shared by two grids. To our best knowledge, SeGrid is the first work that targets key establishment and energy conservation simultaneously. This is a more practical consideration since sensors may stay in sleep mode most of the time for network lifetime extension. We will explore new instantiation ideas for better security provisioning.

As another future research we will explore the applicability of ID-based cryptosystems [35] to SeGrid. In an ID-based encryption system, the public key can be any string (e.g., an email address), and the private key needs to be computed from the public key and other system parameters. The idea of using the grid ID as a public key in SeGrid is very attractive since public key management can be totally avoided.

ACKNOWLEDGMENT

The research of Dr. Xiuzhen Cheng is supported by the NSF CAREER Award no. CNS-0347674.

REFERENCES

[1] W. Liu and Y. Fang, "SPREAD: enhancing data confidentiality in mobile ad hoc networks," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, vol. 4, pp. 2404–2413, HongKong, March 2004.

[2] E. Shi and A. Perrig, "Designing secure sensor networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 38–43, 2004.

[3] Y. Zhang, W. Liu, and W. Lou, "Anonymous communications in mobile ad hoc networks," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, Miami, Fla, USA, March 2005.

[4] D. W. Carman, P. S. Kruus, and B. J. Matt, "Constraints and approaches for distributed sensor network security," Tech. Rep. 00-010, NAI Labs, September 2000.

[5] F. An, X. Cheng, M. Rivera, J. Li, and Z. Cheng, "PKM: a pairwise key management scheme for wireless sensor networks," in *International Conference on Computer Networks and Mobile Computing (ICCNMC '05)*, Zhangjiajie, China, August 2005.

[6] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 197–213, Berkeley, Calif, USA, May 2003.

[7] W. Du, J. Deng, Y. Han, and P. K. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 42–51, Washington, DC, USA, October 2003.

[8] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, vol. 1, p. 597, Hong Kong, March 2004.

[9] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS '02)*, pp. 41–47, Washington, DC, USA, November 2002.

[10] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 52–60, Washington, DC, USA, October 2003.

[11] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Establishing pairwise keys for secure communication in ad hoc networks: a probabilistic approach," in *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP '03)*, pp. 326–335, Atlanta, Ga, USA, November 2003.

[12] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pp. 70–84, Rome, Italy, July 2001.

[13] R. Blom, "An optimal class of symmetric key generation systems," in *Proceedings of the Workshop on Theory and Application of Cryptographic Techniques (EUROCRYPT '84)*, vol. 209, pp. 335–338, Paris, France, April 1985.

[14] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "An energy efficient coordination algorithm for topology maintenance in ad hoc wireless networks," in *Proceedings of the ACM SIGMOBILE Annual International Conference on Mobile Computing and Networking*, pp. 85–96, Rome, Italy, July 2001.

[15] L. Ma, Q. Zhang, and X. Cheng, "A Power Controlled Interference Aware Routing Protocol for Dense Multi-Hop Wireless Networks," submitted.

[16] A. Cerpa and D. Estrin, "ASCENT: adaptive self-configuring sensor networks topologies," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, vol. 3, pp. 1278–1287, New York, NY, USA, June 2002.

[17] E. J. Christine, M. S. Krishna, P. Agrawal, and J. C. Chen, "A survey of energy efficient network protocols for wireless networks," *Wireless Networks*, vol. 7, no. 4, pp. 343–358, 2001.

[18] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and*

Communications Societies (INFOCOM '02), vol. 3, pp. 1567–1576, New York, NY, USA, June 2002.

[19] M. Ding, D. Chen, K. Xing, and X. Cheng, "Localized fault-tolerant event boundary detection in sensor networks," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, vol. 2, pp. 902–913, Miami, Fla, USA, March 2005.

[20] M. Ding, D. Chen, A. Thaeler, and X. Cheng, "Fault-tolerant target detection in sensor networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '05)*, vol. 4, pp. 2362–2368, New Orleans, La, USA, March 2005.

[21] H. Chan and A. Perrig, "PIKE: peer intermediaries for key establishment in sensor networks," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, vol. 1, pp. 524–535, Miami, Fla, USA, March 2005.

[22] D. Liu, P. Ning, and W. Du, "Group-based key pre-distribution in wireless sensor networks," in *ACM Workshop on Wireless Security (Wise '05)*, pp. 11–20, Cologne, Germany, September 2005.

[23] L. Zhou, J. Ni, Chinya, and V. Ravishankar, "Efficient key establishment for group-based wireless sensor deployments," in *ACM Workshop on Wireless Security (Wise '05)*, pp. 1–10, Cologne, Germany, September 2005.

[24] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences," in *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '92)*, E. F. Brickell, Ed., vol. 740 of *Lecture Notes in Computer Science*, pp. 471–486, Santa Barbara, Calif, USA, August 1992.

[25] W. Du, R. Wang, and P. Ning, "An efficient scheme for authenticating public keys in sensor networks," in *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '05)*, pp. 58–67, Urbana-Champaign, Ill, USA, May 2005.

[26] L. Ma, F. Liu, X. Cheng, F. An, and J. Li, "iPAK: an in-situ pairwise key bootstrapping scheme for wireless sensor networks," submitted to ACM MOBIHOC 2006.

[27] F. Liu and X. Cheng, "SBK: a self-configuring framework for bootstrapping keys in sensor networks," submmitted to ACM MOBIHOC.

[28] A. J. Menezes, P. C. V. Oorscho, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, Fla, USA, 2001.

[29] X. Cheng, A. Thaeler, G. Xue, and D. Chen, "TPS: a time-based positioning scheme for outdoor wireless sensor networks," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, vol. 4, pp. 2685–2696, Hong Kong, March 2004.

[30] F. Liu, X. Cheng, D. Hua, and D. Chen, "TPSS: a time-based positioning scheme for sensor networks with short range beacons," in *Proceedings of the International Conference on Computer Networks and Mobile Computing (ICCNMC '05)*, pp. 33–42, Zhangjiajie, China, August 2005.

[31] A. Thaeler, M. Ding, and X. Cheng, "iTPS: an improved location discovery scheme for sensor networks with long-range beacons," *Journal of Parallel and Distributed Computing*, vol. 65, no. 2, pp. 98–106, 2005, Special issue on theoretical and algorithmic aspects of sensor, Ad Hoc wireless, and Peer-to-Peer networks.

[32] A. Boukerche, X. Cheng, and J. Linus, "Energy-aware data-centric routing in microsensor networks," in *Proceedings of the 6th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '03)*, pp. 42–49, San Diego, Calif, USA, September 2003.

[33] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[34] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM '00)*, pp. 243–254, Boston, Mass, USA, August 2000.

[35] D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing," in *Proceedings of the 21st Annual International Cryptology Conference (CRYPTO '01)*, pp. 213–229, Santa Barbara, Calif, USA, August 2001.

Xiuzhen Cheng is an Assistant Professor in the Department of Computer Science at The George Washington University. She received her M.S. and Ph.D. degrees in computer science from the University of Minnesota—Twin Cities in 2000 and 2002, respectively. Her current research interests include wireless and mobile computing, sensor networks, wireless security, statistical pattern recognition, approximation algorithm design and analysis, and computational medicine. She is an Editor for the International Journal on Ad Hoc and Ubiquitous Computing and the International Journal of Sensor Networks. She has served as a TPC Member for various professional conferences such as IEEE INFOCOM '05, IEEE MASS '04, 05, QShine '04, 05, IEEE VTC '03, and so forth. She received the NSF CAREER Award in 2004.

Fang Liu received her B.S. degree from the University of Science and Technology of China in 2000, and her M.S. degree from the Institute of Computing Technology, Chinese Academy of Sciences in 2003, both in computer science. Since Fall 2003, she has been a Ph.D. student in the Department of Computer Science at The George Washington University. Her current research interests include wireless security, localization and tracking, and information retrieval in sensor networks.

Fengguang An obtained his B.S. and M.S. degrees from the National University of Defense Technology of China in 1993 and 1998, respectively. He has been a doctoral student at the Institute of Computing Technology, Chinese Academy of Sciences since September 2002. His current research interests include key management and secure localization in sensor networks, and network security in general.