A Self-Configured Key Establishment Scheme for Large-Scale Sensor Networks

Fang Liu
Department of Computer Science
The George Washington University
Washington, DC 20052, USA
Email: fliu@gwu.edu

Xiuzhen Cheng
Department of Computer Science
The George Washington University
Washington, DC 20052, USA
Email: cheng@gwu.edu

Abstract— **Symmetric key agreement is significant to security provisioning in sensor networks with resource limitations. A number of pairwise key predistribution protocols have been proposed, but the performance is often constrained by the unavailability of topology information before deployment and the limited storage budget within sensors. This paper proposes SBK, a self-configured scheme for bootstrapping keys in large-scale sensor networks. SBK is topology-adaptive, which requires no preloaded keying information but lets sensors compute shared keys with their neighbors after deployment. By removing the randomness inherent to key predistribution schemes, SBK achieves high connectivity (can be 100%) with small storage overhead. An improved scheme, iSBK, is also proposed to speed up the bootstrapping procedure. To the best of our knowledge, SBK and iSBK are the only pure in-situ key establishment protocols that simultaneously achieve good performance in scalability, connectivity, storage overhead, and resilience.**

I. Introduction

Sensor networks are vulnerable to a variety of attacks [19] due to the broadcast nature of wireless communications and the unattended operation in harsh environments. The application of public crypto based systems such as PKI and KDC are very limited because of sensor node constraints (batter supply, CPU, memory, etc.) and networking constraints (wireless, ad hoc, etc.) [6]. Researchers have proposed different probabilistic-based schemes relying on preloading key-related information (termed *key predistribution*) within each sensor for bootstrapping shared keys between neighboring sensors after deployment (Eg. [7]–[13], [20], etc.) such that symmetric cryptography can be applied. These schemes have been evaluated mainly in terms of *scalability in network size* (hereafter referred to as *scalability*), *connectivity of the key-sharing graph* (hereafter referred to as *connectivity*), *storage overhead*, and *resilience against node capture attack* (hereafter referred to *resilience*). A good key establishment scheme is expected to have high scalability, high connectivity, high resilience, and low storage overhead. However, none of the existing schemes achieves all these four goals[1].

The two extreme cases for key predistribution are the *single master key scheme* and *all pairwise keys scheme*. A single master key shared by all sensors is inadequate since a successful attack on one node compromises the whole network. This scheme has the most efficient usage of memory and has good scalability. On the other hand, the all pairwise keys scheme in which a unique key exists for every pair of sensors is not appropriate for large-scale sensor networks due to the high memory overhead and the complicated management. However, this scheme is *perfect* in resilience because the compromise of one sensor does not affect the security of the links maintained by un-compromised nodes. These two schemes achieve "perfect" connectivity in the key-sharing graph.

Other than these two extreme schemes we have the *random keys scheme*, the *random pairwise keys scheme*, the *random key spaces scheme*, and the *group-based scheme*, which explore the tradeoff among scalability, connectivity, storage, and resilience. The *random keys scheme* [7], [11] loads

[1]Communication and communication overheads are important metrics too but compared to these four parameters they are considered secondary because most key bootstrapping schemes rely on local broadcasting and simple operations for shared key discovery

a random key ring drawn from a large key pool to each sensor and two neighboring sensors establish a shared key if they have common keys in their key rings. Schemes in this category exhibit a conflict between the key pool size, and the resilience and connectivity when the key ring size is constrained to the storage budget. A larger key pool results in higher resilience but lower connectivity. Further, increasing the number of captured sensors increases the percentage of compromised links shared by un-captured sensors.

Random pairwise keys schemes [7], [8] overcome the last drawback of the random keys scheme, but still could not solve the conflict between the key pool size, and the resilience and connectivity. In this category, a random key is shared by only two nodes, with the node pairing either ID-based [7], or virtual grid/group-based [8]. *Random key spaces schemes* [9], [12] preload crypto shares from multiple key spaces to each sensor, and two neighboring sensors establish a shared key after deployment if and only if they have crypto shares from the same key space. Compared to the previous two categories, random key spaces schemes have better scalability due to the exploitation of multiple key spaces. But the improvement in connectivity and resilience is still constrained by the limited storage within each sensor. *Group-based schemes* [10], [13], [20] divide sensors into disjoint groups. Each sensor contains information for intra-group key establishment and inter-group key establishment. Compared to the previous three categories, schemes in this category have better performance in terms of scalability, connectivity, resilience, and storage overhead. But the conflict remains since each sensor still needs to be preloaded with information that may never be used by the sensor.

Note that the unavailable topology information before deployment renders all key predistribution schemes fail in achieving all the four objectives of key establishment in sensor networks, namely scalability, connectivity, storage, and resilience. One major reason is the *"randomness"*[2] involved in the predistribution schemes, since sensors are preloaded with a variety of key-related information that may never be used after deployment. SBK, a self-configuring framework for bootstrapping keys in large-scale sensor networks proposed in this paper, achieves all these four goals simultaneously.

[2]With "randomness", we refer to the fact that key predistribution schemes preload more information than needed such that two neighboring sensors can establish a shared key with higher probability after deployment.

SBK is topology-adaptive. Sensors differentiate their roles as either service nodes or worker nodes after deployment by probing the local connectivity of the network. Service sensors construct key spaces, and distribute keying information in order for worker sensors to bootstrap pairwise keys. By exploring the λ-collusion resistent property of the key spaces, SBK achieves perfect security in against node capture attack. Further, SBK has low storage overhead but can establish secure communication for almost all pairs of neighboring sensors. SBK is a localized procedure, which preloads no random information to any sensor before deployment, and thus achieves a perfect scalability in network size. We also propose an improved SBK (iSBK) to speed up the bootstrapping procedure. iSBK retains all the nice features of SBK but achieves even better connectivity with the tradeoff of a slight increase in storage and computation overhead.

Compared to existing schemes proposed for shared key construction in sensor networks, SBK and iSBK have the following characteristics and advantages:

- SBK/iSBK is a topology-adaptive localized scheme that requires no *a priori* knowledge of post-deployment configuration. This feature is particularly attractive since in many applications sensors are dropped from aircrafts and the topology is unpredictable.
- SBK/iSBK has high scalability in network size; The induced key-sharing graph has high connectivity; Each sensor has low storage overhead; And SBK/iSBK has perfect resilience against node capture attack.

The remainder of the paper is organized as follows. Preliminaries, models, and assumptions are sketched in Section II. SBK, the self-configured key establishment scheme is proposed in Section III and evaluated in Section IV. An improved version (i.e. iSBK) is elaborated in Section V. Simulation study for both schemes are reported in Section VI. Finally, we conclude our paper in Section VII.

II. Preliminaries, Models, and Assumptions

A. Key Space Models

Our scheme works fine with both key space models introduced by [3], [4]. The two models are similar in nature, both ensure the λ-collusion resistance[3], and have been tailored for sensor networks by [12] and [9], respectively.

The polynomial-based key space utilizes a symmetric λ-degree polynomial $f(x,y) = \sum_{i,j=0}^{\lambda} a_{ij}x^j y^j$ over a finite field F_q, where q is a prime that is large enough to accommodate a cryptographic key. A sensor i is preloaded with its crypto share $f(i,y)$. After exchanging the node IDs, two sensors i and j can compute a shared key from their crypto shares as $f(i,j) = f(j,i)$.

The matrix-based key space utilizes a $(\lambda+1) \times (\lambda+1)$ public matrix[4] G and a $(\lambda+1) \times (\lambda+1)$ private matrix D over a finite field $GF(q)$, where q is a large prime number. Let $A = (D \cdot G)^T$. D is symmetric, then $K = A \cdot G$ is symmetric too. Thus we have $k_{ij} = k_{ji}$, where k_{ij}, the element at the ith row and the jth column of K, can be computed from the ith row of A and jth column of G. A sensor i is allocated a crypto share containing the ith row of A and the ith column of G. Two sensors i and j can compute their shared key k_{ij} by exchanging the columns of G in their crypto shares.

B. Rabin's Public Cryptosystem

We need Rabin's public cryptosystem [16] as a crypto primitive to establish a computationally asymmetric secure channel through which crypto shares can be delivered from a service sensor to a worker sensor. Rabin's system requires a public key n and a private key (p,q) such that $n = p \cdot q$, where p and q are large primes. The encryption of a message M, denoted by $E_n(M||B) = (M||B)^2 \bmod n$, involves one modular squaring operation, where B is a predefined pattern for ambiguity resolution in Rabin's decryption. However, recovering the plaintext M by computing $D_{p,q}(E_n(M||B))$ takes much higher computation (comparable to that of RSA).

[3]In a λ-collusion resistent key space, as long as no more than λ sensors are compromised, the keying information within the remaining nodes are still kept perfectly secure.

[4]Note that G can contain more than $(\lambda+1)$ columns.

C. Network Model and Security Assumptions

We consider a large-scale sensor network with homogeneous sensors dropped over the deployment region through vehicles such as aircrafts. Therefore no topology information is available before deployment. We assume that sensors are coarsely time-synchronized before deployment such that they can start the bootstrapping procedure roughly simultaneously.

In our consideration, sensors have the same configuration (e.g. communication capability, resources, preloaded information, etc) before deployment. After deployment, they take different roles and self-configure accordingly. Some nodes become service nodes and take the responsibility of security information computation and distribution. Others self-configure as worker nodes and are in charge of sensing and reporting data.

We assume initial trust exists among sensors within a short period of time after deployment. An adversary can only passively monitor a small proportion of the communications during the node self-configuration time, while no active attacks (flooding, jamming, spoofing, etc.) can be launched and no physical access to the network can be obtained during this initial period. This assumption is realistic, since a sensor deployed in a security-critical environment must be designed to survive at least a short interval $T_{survival}$ when captured by an adversary[5] [2] [21]; otherwise, the whole network can be easily taken over by the opponent. But after the initial bootstrap procedure, an adversary is capable of all possible attacks. Once a node is compromised, all the information it holds will be released.

III. SBK: The Self-configured Key Establishment Scheme

SBK consists of four phases. Sensors are preloaded with a bootstrap program and several system parameters in the *predistribution* phase, and differentiate their roles as either service nodes or worker nodes in the *node self-configuration* phase. A worker sensor obtains keying information through a computationally asymmetric secure channel from a service node in the *crypto share distribution* phase, and then computes keys shared

[5]$T_{survival}$ is the minimum time needed to reverse-engineer a sensor.

with those nodes belonging to the same key space in the *shared key discovery* phase. *After the bootstrapping procedure terminates, all sensors erase their key space information for security reasons.*

A. Predistribution

Before deployment, each sensor is preloaded with a bootstrapping program that controls the role differentiation and node configuration. For a sensor that decides to become a service node, this program also computes a key space and two prime numbers for securely disseminate crypto shares to worker nodes in the vicinity.

Several pre-configured system parameters, listed in Table I, are also uploaded to each sensor. k_0, a key shared by all sensors, is employed to secure all the messages exchanged during the bootstrapping procedure. λ, the security parameter, determines the maximum number of worker nodes to be served by a service sensor. H, the forwarding bound, is the maximum distance in hop-count over which the existence of a key space can be announced. The probability of service node election is denoted by P_s. The initial values of H and P_s are determined by λ according to the following criteria:

$$D_H \leq \lambda, \quad (1)$$

$$P_s = 1/\lambda, \quad (2)$$

where D_H is the average number of neighbors within H-hop distance in the network. Note that D_H can be estimated based on the deployment area, the nominal transmission range of a sensor node, and the number of sensors to be deployed in the network. The time per round for service node election is denoted by T_s while the total time for sensor self-configuration is T_{sbk}. For simplicity, we assume $T_{sbk} = t \cdot T_s$, where t is the total number of rounds.

k_0	The shared key by all sensors to secure the bootstrap procedure
λ	Security parameter. Also specifies the maximum number of nodes served by a service sensor
H	The forwarding bound of key space advertisement
P_s	The probability of service node election
T_s	The time of one round for service node election
T_{sbk}	The total time for node self-configuration

TABLE I
PRELOADED SYSTEM PARAMETERS.

B. Node Self-Configuration

Right after deployment, a sensor bootstraps and elects itself as a service node with the probability P_s. Whenever a node decides to become a service one, the bootstrapping program generates a key space (see Subsection II-A) consisting of λ crypto shares. Two prime numbers p and q are also computed for Rabin's algorithm to establish a computationally asymmetric secure channel for crypto share distribution from service sensors to worker sensors (to be explained in Subsection III-C).

The service node election process is repeated every T_s-time for t rounds. At each round, a non-service sensor that does not have any service node within $H-1$ hops chooses to become a service node with the probability P_s. If a sensor succeeds in the self-election, it sets up a key space and enters the next phase for crypto share distribution. Otherwise, it listens to key space advertisements. Upon receiving any new key space announcements from a service node that is at most $H-1$ hops away (to be explained in Subsection III-C), the sensor becomes a worker node and enters the next phase for crypto share acquisition. If no service node within $H-1$ hops is detected in the current round, the sensor participates in the next round.

C. Crypto Pair Distribution

In the third phase, a public key assisted *Crypto Pair Distribution Protocol* (*CPD*) is used to securely disseminate crypto shares from a service sensor to the worker sensors in the neighborhood. After the key space is set up, a service node broadcasts a beacon message notifying others of its existence. A worker node receiving the beacon then requests a crypto share from the service node. To transmit the crypto share safely, a secure channel based on Rabin's system is established between the two sensors. CPD is composed of the following three steps:

1) Key Space Advertisement: A service node announces its existence through beacon broadcasting when its key space is ready. As illustrated by Fig. 1, the beacon message includes: a) a unique key space ID, b) the public key n, where $n = p \times q$ and (p, q) is the corresponding private key generated by the bootstrap program, and c) a TTL value that is initialized to be the forwarding bound H. Nodes receiving the message first subtract 1 from TTL, then forward it if the new TTL value is larger than zero. Hence, any sensor receiving the beacon message is at most H-hop distance away from the source service node. A worker node sets up a secure channel to request a crypto share after it receives the key space existence notification.

id	*n*	*TTL*

id: id of the service node *n*: the public key, *n=p x q*
TTL: time to live, initialized as the forwarding bound *H*

Fig. 1. The message format of key space advertisement

As stated in the node self-configuration phase, a sensor must decide whether it is eligible for service node election in the next round. Actually, this can be realized easily based on the format of the message given in Fig. 1. If a sensor receives any beacon with $TTL \geq 1$, then at least one service node exists within $(H-1)$-hop distance. In this case, the node turns out to be a worker sensor and terminates its self-configuration process. Otherwise, the sensor intends to become a service node with the probability P_s in the next round.

2) Secure Channel Establishment: Any worker node requesting a crypto share from a service node needs to establish a secure channel to the associated service node. Recall that we leverage Rabin's public key cryptosystem [16] for this purpose. The public key n is announced through the beacon broadcasting at the previous step. Therefore a worker sensor can pick up a random number k and compute $E_n(k||B) = (k||B)^2 \; mod \, n$. $E_n(k||B)||B$ is transmitted to the corresponding service node, which computes $D_{p,q}(E_n(k||B))$ by applying the decryption algorithm[6]. Now k is known to both the worker node and the service node, and can be used as the secret key of a secure channel for the crypto share dissemination in the next step.

As stated in Subsection II-B, Rabin's cryptosystem is asymmetric in the computational complexity of encryption and decryption. Thus, the secure channel establishment shifts a large amount of the computation overhead to service nodes. Note that worker sensors are expected to operate for years while service nodes can die after their duty is complete. In this aspect, service nodes work as sacrifices to extend the network's lifetime.

[6]Note that B is used for ambiguity resolution in Rabin's decryption. Therefore it is transmitted as a plain text.

3) Crypto Share Acquisition: After a shared key k is established between a worker node and a service node, the service node can allocate to the worker node a crypto share from its key space. Note that a service sensor can only assign crypto shares on request to at most λ worker nodes. The crypto share, encrypted with k based on any popular symmetric encryption algorithm (AES, DES, etc), is transmitted to the requesting worker node securely. Any two worker nodes receiving crypto shares from the same service node can compute a shared key for secure data exchange in the future.

D. Shared Key Discovery

Two neighboring nodes sharing at least one key space (having obtained crypto shares from at least one common service node) can establish a shared key accordingly. As stated in Subsection II-A, the method to compute a shared key is dependent on the underlying key space model. Note that this procedure involves the exchange of either node IDs, if polynomial-based key space model is utilized [4], or columns (seeds) of the public matrix, if matrix-based key space model is utilized [3]. To further improve security, nonces can be introduced in against replay attacks.

IV. Evaluation on SBK: Analytical Results

We evaluate the security of SBK in terms of resilience against node capture attack and connectivity of the induced key-sharing graph, and measure the performance in terms of storage, computation and communication overheads. Since service nodes are designed as sacrifices that do not obviously affect the lifetime of a large-scale sensor network, we care about the performance of worker nodes only.

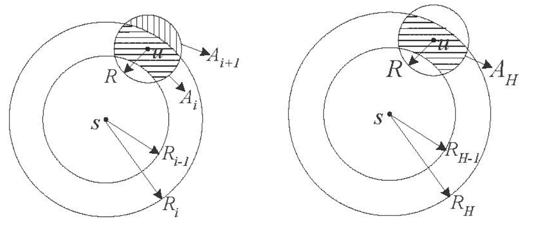
A. Connectivity of the Key-Sharing Graph

The effectiveness of a key establishment scheme is dependent on the connectivity of the final key-sharing graph $G_K(V, E)$, where V is the set of sensors in the network, and E is the set of edges incident to two nodes that can securely communicate (i.e. have shared keys). The probability that two neighboring sensors have a shared key is the *local connectivity* P_{local}, and the probability that the key-sharing graph G_K is connected is the *global connectivity* P_{global}. According to Erdős and Rényi's connectivity theory [18], P_{global} can be uniquely determined by P_{local}, given the key establishment algorithm and a sensor network with average node degree d. Therefore, we evaluate SBK in terms of P_{local} only.

Note that P_{local} is defined to be the probability that two neighboring sensors share at least one common key space. To study P_{local}, we need the probability that two neighboring sensors u and v obtain crypto shares from the same service node s, i.e. u, v are both within H hops from s. Since u, v are immediate neighbors, only two possible cases exist:

- Case 1: u is an ith-hop neighbor of s, and v is an ith-hop or (i+1)th-hop neighbor of s, where i=1, 2, ... , $H-1$.
- Case 2: u, v are both Hth-hop neighbors of s.

Computing the probability of either Case 1 or Case 2 is a very challenging problem. In our study, we exploit the *Effective Radius* (ER) model derived in [14], which computes an effective radius R_h to enclose all the h-hop neighbors within a disk region. As validated by [14], the ER model is an effective approximation when the network is uniformly and densely deployed.



(a) Case 1: v is in area A_i or A_{i+1} (b) Case 2: v is in area A_H

Fig. 2. Two neighboring nodes belong to the same key space.

From the ER model, all ith-hop neighbors of s reside in the non-overlapping area of the two disks defined by R_{i-1} and R_i, as indicated by Fig. 2. Therefore for Case 1, v is either in the area A_i, or in the neighbor area A_{i+1}, as shown in Fig. 2(a). For Case 2, v can only be in the area A_H, as shown in Fig. 2(b).

Let p_A be the probability that a node is within a given area A. Since the nodes are uniformly distributed in the network, p_A can be estimated with the coverage area of A. The probability p_0 that the neighboring nodes u, v share the key space of the service node s can thus be estimated as:

$$\begin{aligned} p_0 &= Pr[u \in K_s \text{ and } v \in K_s | d_{u,v} \leq R] \\ &= \sum_{i=1}^{H-1} \frac{d_i}{N}(p_{A_i} + p_{A_{i+1}}) + \frac{d_H}{N} \cdot p_{A_H} \end{aligned} \quad (3)$$

where d_h is the number of nodes that are h-hop distance away, K_s is the key space provided by service node s, $d_{u,v}$ denotes the distance between node u and v, and R is the nominal transmission range. Hence, the probability that any two neighboring nodes sharing at least one common key space, denoted as P_{local}, is:

$$P_{local} = 1 - (1 - p_0)^{\sum_{i=1}^{t} N_s^i} \quad (4)$$

where N_s^i is the number of service nodes generated in the i-th round computed by Eq. (7), and $t = T_{sbk}/T_s$ is the total number of rounds.

B. Storage Overhead

For simplicity, we count the storage overhead of our protocol as the expected number of crypto shares (either polynomial shares [12] or matrix shares [9]) each worker sensor receives from the surrounding service sensors. The exact amount of memory consumption can be easily derived with known key space information.

Assume N sensors are uniformly distributed in the network. Each sensor is pre-configured with the following parameters: λ, H, P_s, T_s and T_{sbk}, as defined in Table I. In the first round, the number of service nodes generated is:

$$N_s^1 = N \cdot P_s, \quad (5)$$

Among all the remaining sensors, those having no service node within $H-1$ hops can participate in the second round for service node election. We denote the number of sensors within h hops in the neighborhood by D_h. Thus the number of service nodes generated in the second round is as follows:

$$N_s^2 = (N - N_s^1) \cdot (1 - P_s)^{D_{H-1}} \cdot P_s, \quad (6)$$

Similarly, a node can elect itself as a service node with probability P_s in the j-th round if and only if all of its neighbors within $H-1$ hops fail in the first $j-1$ rounds of elections. Therefore,

$$N_s^j = (N - \sum_{i=1}^{j-1} N_s^i) \cdot ((1 - P_s)^{(j-1)})^{D_{H-1}} \cdot P_s \quad (7)$$

Altogether, there are $t = T_{sbk}/T_s$ rounds of service node election. Since each service node can provide key information to at most λ worker sensors, the average number of keys stored within each worker sensor can be estimated by[7]:

$$\tau \approx \lambda \times \frac{\sum_{i=1}^{t} N_s^i}{N - \sum_{i=1}^{t} N_s^i} \tag{8}$$

C. Resilience against Node Capture Attack

Recall that in our SBK scheme a service node can distribute crypto shares to at most λ worker sensors and then delete all the key space information, and our key spaces are λ-collusion resistent (Subsection II-A), it is impossible for an attacker to compromise any key space. Further, if all key spaces are unique, all shared keys by neighboring sensors are unique. This is almost certainly true in SBK since each service node constructs the key space randomly and independently. Therefore links protected by a shared key based on SBK remain secure as long as none of the two end sensors is compromised. This is a dramatic improvement over Random Keys schemes ([7] [11], etc.), in which a key may be shared by multiple links.

A reader may argue that the limitation on the number of worker nodes sharing one key space may lead to a large number of service nodes to be generated in SBK, since each worker node should have at least one key to maintain the connectivity of the key-sharing graph. Actually this represents a trade-off between security and cost. In many cases, it is desirable to sacrifice a small fraction of low-cost sensors to achieve higher level of security. We will further study this by simulation in Section VI.

D. Computation and Communication Overheads

In SBK, the computation overhead of a worker node comes from three sources: encrypting a shared key k between a service node and itself in *secure channel establishment*, decoding the keying information obtained from the associated service node in *crypto share acquisition*, and calculating the pairwise keys shared with the neighbors in *shared key discovery*. The first involves one modular squaring while the second requires a symmetric decryption operation. On average a worker node completes τ (see Eq. (8)) such operations.

[7]The actual average is expected to be smaller than τ.

In general, given the crypto shares, computing a shared key with one neighbor takes $(\lambda + 1)$ modular multiplications for both key space models. This must be repeated $d \times P_{local}$ times per sensor on average, where d is the average node degree. However, if the matrix-based key spaces are used, and only a seed instead of the whole column of the public matrix G is included in the crypto share, each worker sensor needs $(\lambda - 1) \times \tau$ more modular operations for recovering the complete crypto shares.

The communication overhead of the worker sensors in SBK results from requesting crypto shares from service nodes, and relaying messages for others. Each worker sensor also needs to exchange information with its neighbors for shared key computation. The number of broadcastings per sensor is bounded by $O(\lambda \cdot \tau + d)$. Actually this is overestimated since in reality a sensor does not need to relay messages for all the worker sensors with which to share a key space.

V. iSBK: The Improved SBK Scheme

For security purpose, we expect that the time for the key bootstrapping procedure won't last too long, i.e. we expect that most of the worker nodes can obtain crypto shares from service nodes and establish shared keys with their neighbors within a short time.

Given the security parameter λ and the duration time T_s per round for service node election and configuration, we would like to achieve high connectivity in the final key-sharing graph G_K while minimizing the whole configuration time T_{sbk}. According to the theoretical analysis in Section IV, the connectivity of G_K is dependent on t, where $t = T_{sbk}/T_s$, and the number of service node N_s^i generated in each round i for a given network (see Eqs. (3)(4)). As indicated by Eq. (7), we must increase P_s to decrease t. In the basic SBK scheme, P_s is fixed to be $1/\lambda$. Since each sensor is expected to be associated with at least one key space, $1/\lambda$ is just the lower bound to ensure the connectivity in the final key sharing graph. Therefore, node configuration may take a relatively long time in the basic scheme.

In this section, we propose an improved scheme (iSBK) that can achieve a higher connectivity in a shorter time. Similar to SBK, iSBK contains four phases. The differences reside in the first two stages:

Predistribution: In iSBK, we modify the initial values for the forwarding bound H of key space advertisement and the probability of service node election according to the following criterions:

$$D_H \leq \lambda, \quad (9)$$
$$P_s = 1/D_{H-1}, \quad (10)$$

where D_h is the expected number of nodes within h-hop neighborhood, as defined before.

Node Self-Configuration: In this phase, the possibility of a node elected as a service node is doubled in each new round until it reaches 1.

The iSBK scheme generates more service nodes than the basic algorithm. As mentioned earlier, the motivation for doing so is to reduce the configuration time in order to minimize the danger of "exposing" sensors insecurely to adversaries. However, due to the low cost of sensors, the number of service nodes as sacrifices should be tolerable. In the next section, we evaluate the cost of SBK and iSBK in terms of the percentage of service nodes to be generated in a network through simulation study. The results indicate that iSBK can achieve a better performance with a small increase in the number of service nodes involved.

VI. Evaluation on SBK and iSBK: Simulation Results

We evaluate SBK and iSBK through simulation study in this section. For all of the following experiments, we consider a sensor network deployed over a field of 100 by 100. A number of sensors are uniformly distributed[8], and each node is capable of a fixed transmission range of 10. All the results are averaged over 100 runs.

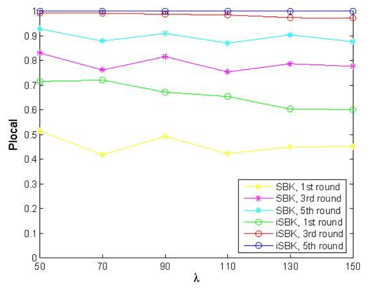
Note that the security of SBK has been studied in Subsection IV-C. SBK can protect the secrets intact among uncaptured nodes no matter how many nodes are captured by adversaries and no matter what the network topology will be. iSBK possesses the same nice property since it follows the same rule that there are at most λ worker nodes within each λ-collusion resistent key space.

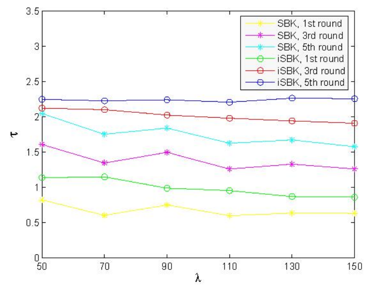
[8]Though only uniform distribution is studied in the simulation, both SBK and iSBK can be applied to any network deployment.

In the following, we evaluate SBK and iSBK in terms of *storage*, measured by τ, the number of keying information units (polynomial shares [4] or crypto shares [3]) obtained by a worker node; P_{local}, measured by the fraction of communication links that are secured by shared keys; and *cost*, measured by the percentage of service nodes generated. In SBK (iSBK), three system parameters affect the performance, i.e. the node density of the network, the security parameter of the λ-collusion resistent key spaces, and the number of rounds for service node election ($t = T_{sbk}/T_s$). We design two experiments to study the impact of these factors.

In the first experiment, we deploy 300 sensors in the region. Fig. 3 shows the simulation results obtained after the $1st$, $3rd$ and $5th$ round of service node election. We notice that iSBK achieves a higher level of connectivity within a much shorter time than the basic scheme. As illustrated in Fig. 3(a), $t = 3$ is enough for iSBK to achieve P_{local} close to 1, while using SBK, only 80% of the neighboring pairs can establish secure communication with the same amount of time. We also notice that compared to the basic scheme, iSBK generates more service nodes, and therefore worker sensors carry more key spaces, as indicated by Figs. 3(b), 3(c). Nonetheless, in iSBK, each worker node still consumes only a small amount of memory ($\tau \approx 2$) to achieve high connectivity. Note that both schemes achieve a desirable level of connectivity at the expense of a very small storage overhead in worker nodes. For example, almost all worker nodes can establish secure communication with their neighbors using iSBK, with each worker sensor storing about 2 crypto shares when $t = 3$. Even with the basic SBK, P_{local} is above 90%, with each worker node carrying less than 2 key spaces when $t = 5$. In summary, both SBK and iSBK can achieve very good connectivity and conserve the resources of worker nodes effectively by selecting some service nodes as sacrifices. Between the two schemes, iSBK requires a shorter configuration time with a reasonable storage overhead in worker nodes.

In the second experiment, we study the performance of SBK and iSBK under different network densities with t fixed to 3. We deploy 300 or 500 nodes in the network area. As illustrated in Fig. 4, iSBK still outperforms the basic one while both achieve similar connectivity under different node densities. In a denser network, both schemes


(a) P_{local}: the local connectivity


(b) τ: the keying information storage

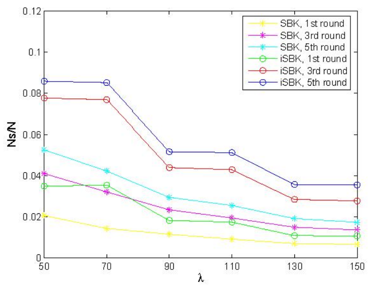
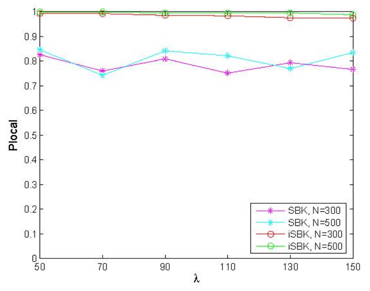
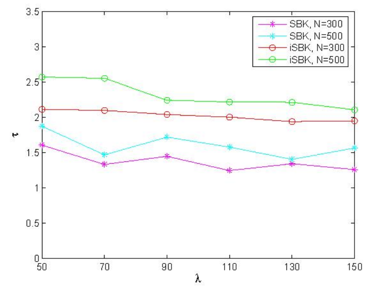

(c) N_s/N: the percentage of service nodes

Fig. 3. Test 1. Performance of the basic and the improved SBK schemes: in the $1st$, $3rd$ and $5th$ round, with $N = 300$.


(a) P_{local}: the local connectivity


(b) τ: the keying information storage

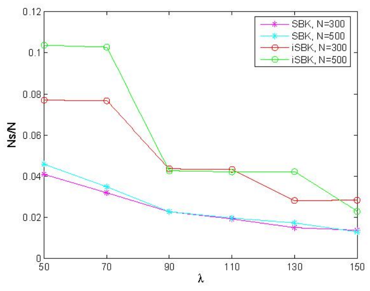

(c) N_s/N: the percentage of service nodes

Fig. 4. Test 2. Performance of the basic and the improved SBK schemes: t =3, with different network density.

generate a larger number of service nodes and incur heavier storage overhead on each worker node under a given λ, since one service node can only serve at most λ worker sensors.

We observe that SBK exhibits a "smoother" curve in Figs. 3(c), 4(c), while iSBK generates similar percentage of service sensors for $\lambda = 50$, 70, and for $\lambda =$ 90-130. The reason is that for iSBK, different λ values may result in the same H and P_s pair according to Eqs. (9)(10). Specifically, in our experiments, $\lambda = 50$ and 70 lead to the same pair of H and P_s, so do $\lambda =$ 90-130.

Furthermore, both experiments also indicate that SBK (iSBK) achieves similar connectivity and storage overhead for different λ. This indicates that our schemes do make sensors "self-configure" according to the environments (system parameters, node density, etc.) and elect service nodes when necessary. According to Eqs. (1)(2), Eqs. (9)(10), a key space is set up with the probability P_s and advertised within the H-hop neighborhood, both defined by λ. The immunity from the variation of λ shows that the selection of P_s and H is appropriate in our schemes. Since λ determines the maximum number of worker nodes that can be served by one service sensor, we can expect a lower cost under a larger λ. This coincides with the results in Figs. 3(c), 4(c), where the number of service nodes decreases as λ increases.

Note that we have also conducted extensive simulation to compare multiple existing key predistribution schemes [7], [9], [11], [12] with SBK and iSBK in terms of connectivity and storage overhead, and have achieved results very favorable to SBK and iSBK. For example, with the same parameter settings for λ and τ, for SBK $P_{local} > 80\%$ while for [9] $P_{local} \approx 15\%$. Due to space limitations, these results have to be reported in a different paper.

VII. Conclusion

In this paper, we propose SBK, a self-configured key establishment scheme for large-scale sensor networks. SBK is fundamentally different compared to all existing key predistribution schemes. It is an in-situ key establishment scheme that achieves high scalability in network size since it is pure localized. It exhibits perfect resilience against node capture attack by exploring the λ-collusion resistent property of the underlying key space models. Since SBK requires no predistribution of random keying information, the randomness inherent to all key predistribution schemes have been completely removed. Therefore SBK achieves high connectivity in the key-sharing graph with low storage overhead. We also propose iSBK, which speeds up the bootstrapping procedure of SBK. To our best knowledge, SBK and iSBK are the first that simultaneously achieve good performance in terms of scalability, connectivity, storage overhead, and resilience.

ACKNOWLEDGMENT

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

REFERENCES

[1] Crossbow MPR400/410/420 MICA2 Mote: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf

[2] R. Anderson, H. Chan, A. Perrig, Key Infection: Smart Trust for Smart Dust, in *ICNP'04*, pp. 206-215, 2004.

[3] R. Blom, An optimal class of symmetric key generation systems, In *EUROCRYPT 84*, pp. 335-338, 1984.

[4] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, Perfectly-secure key distribution for dynamic conferences, in *CRYPTO'92*, LNCS 740, pp. 471-486, 1992

[5] S. A. Camtepe and B. Yener, Key Distribution Mechanisms for Wireless Sensor Networks: a Survey, RPI Technical Report TR-05-07, March 23, 2005.

[6] D. W. Carman, P. S. Kruss, and B. J. Matt, Constraints and approaches for distributed sensor network security, *NAI Labs Technical Report #00-010*, Sept. 2000.

[7] H. Chan, A. Perrig, and D. Song, Random key predistribution schemes for sensor networks, in *S&P'03*, 2003, pp. 197-215.

[8] H. Chan and A. Perrig, PIKE: Peer intermediaries for key establishment in sensor networks, in *Proceedings of IEEE Infocom*, Miami, March 2005.

[9] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, A pairwise key pre-distribution scheme for wireless sensor networks, in *CCS'03*. ACM Press, New York, NY, 2003, pp. 42-51.

[10] W. Du, J. Deng, Y. S. Han, S. Chen, and P. Varshney, A key management scheme for wireless sensor networks using deployment knowledge, in *Proceedings of IEEE Infocom*, Hong Kong, March 2004, pp. 586-597.

[11] L. Eschenauer and V. D. Gligor, A key-management scheme for distributed sensor networks, in *CCS'02*. ACM Press, 2002, pp. 41-47.

[12] D. Liu and P. Ning, Establishing pairwise keys in distributed sensor networks, in *CCS'03: Proceedings of the 10th ACM conference on Computer and Communications Security*. ACM Press, New York, NY, 2003, pp. 52-61.

[13] D. Liu, P. Ning, and W. Du, GroupBased Key PreDistribution in Wireless Sensor Networks, *Wise'05*, 2005.

[14] L. Ma, X. Cheng, and W. Li, The Effective Radius Model: Analysis and Validation, 2005.

[15] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, Inc., 2001.

[16] M. O. Rabin, Digitalized signatures and public-key functions as intractable as factorization, *Technical Report MIT/LCS/TR-212*, Laboratory for Computer Science, MIT, 1979.

[17] R. L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, vol. 26, no. 2, Feb. 1978, pp. 120-126.

[18] J. Spencer, The Strange Logic of Random Graphs, *Algorithms and Combinatorics 22*, Springer-Verlag, 2001.

[19] K. Xing, S. Srinivasan, M. Rivera, J. Li, and X. Cheng, Attacks and Countermeasures in Sensor Networks: A Survey, The George Washington University Technical Report GWU-CS-TR-010-05, 2005.

[20] L. Zhou, J. Ni, C. V. Ravishankar, Efficient Key Establishment for Group-Based Wireless Sensor Networks, *Wise'05*, pp. 1-10, 2005.

[21] S. Zhu, S. Setia, and S. Jajodia, Leap: efficient security mechanisms for large-scale distributed sensor networks, in *CCS'03: Proceedings of ACM Conference on Computer and Communications Security*, 2003, pp. 62-72.