SBK: A Self-Configuring Framework for Bootstrapping Keys in Sensor Networks

Fang Liu, *Member*, *IEEE*, Xiuzhen (Susan) Cheng, *Member*, *IEEE*,
Liran Ma, *Student Member*, *IEEE*, and Kai Xing, *Student Member*, *IEEE*

Abstract—Key predistribution has been claimed to be the only viable approach for establishing shared keys between neighboring sensors after deployment for a typical sensor network. However, none of the proposed key predistribution schemes simultaneously achieves good performance in terms of scalability in network size, key-sharing probability between neighboring sensors, memory overhead for keying information storage, and resilience against node capture attacks. In this paper, we propose SBK, an in situ self-configuring framework for bootstrapping keys in large-scale sensor networks. SBK is fundamentally different from all key predistribution schemes. It does not require keying information[1] predeployment. In SBK, sensors differentiate their roles as either service nodes or worker nodes after deployment. Service sensors construct key spaces and distribute keying information in order for worker sensors to bootstrap pairwise keys. An improved scheme iSBK is also proposed to speed up the bootstrapping procedure. We conduct both theoretical analysis and simulation study to evaluate the performances of SBK and iSBK. To the best of our knowledge, SBK and iSBK are the only in situ key establishment protocols that simultaneously achieve good performance in scalability, key-sharing probability, storage overhead, and resilience against node capture attacks.

Index Terms—Wireless sensor networks, in situ key establishment, key predistribution.

1 Introduction

Secure communication [27] is critical for many sensor network applications. Due to its efficiency, symmetric key cryptography is very attractive in sensor networks. However, bootstrapping shared keys for neighboring sensors is a challenging problem attributed to the unavailability of topology information before deployment and the limited storage budget within sensors[2] [6]. Even though a number of key predistribution protocols have been proposed (for example, [7], [10], [11], [13], [14], [16], and [17]), none of them can simultaneously achieve good performance in terms of *scalability in network size*, *key-sharing probability between neighboring sensors*, *memory overhead for keying information storage*, and *resilience against node*[3] *capture attacks*. In this paper, we propose SBK, an in situ self-configuring framework for bootstrapping keys in large-scale sensor networks.

In SBK, keys are computed dynamically and automatically after deployment based on several *system parameters* pertaining to network density, sensor nominal transmission range, memory budget for security information storage, etc. No deployment knowledge is required beforehand, and nodes explore the particulars of the real network settings themselves and establish pairwise keys accordingly. In SBK, a fraction of sensors self-elect to become *service nodes* if need be. These sensors are in charge of key space generation and distribution. The majority of the sensors, namely, *worker nodes*, get *keying information* from service sensors in their vicinity. A piece of keying information is called a *keying share*. Each service sensor generates a key space and distributes the corresponding keying shares to at most λ worker sensors through a computationally asymmetric channel that shifts the large amount of computation overhead to the service node. Each worker sensor receives at most one keying share from one key space (service sensor). These key spaces are λ-collusion resistant, which means that a key space remains secure as long as not more than λ keying shares (worker sensors) of the key space are released to (captured by) an attacker. Two worker sensors can compute a common key once they obtain keying information from the same service node. SBK regulates that at most λ worker sensors reside in the same key space, leading to a "perfect" resilience against node capture attacks. Furthermore, SBK has low memory overhead but can establish secure communication for almost all pairs of neighboring sensors. SBK is a localized procedure which preloads no random keying 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. The simulation study indicates that iSBK achieves even better key-sharing probability, with the trade-off of a

1. Keying information is defined to be a piece of information drawn from a key space for shared key computation.

2. The conflict between the large memory requirement due to the high keying information redundancy in key predistribution and the limited memory budget is sometimes referred to as "randomness" [12], since the useful information is "random" before deployment.

3. Note that we use sensors, nodes, and sensor nodes interchangeably throughout this paper.

- *F. Liu is with the Department of Computer Science, University of Texas—Pan American, ENGR 3.272, 1201 W. University Drive, Edinburg, Texas 78539-2999. E-mail: fliu@cs.panam.edu.*
- *X. Cheng is with the Department of Computer Science, The George Washington University, 801 22nd Street NW, Phillips Hall, Suite 704, Washington, DC 20052. E-mail: cheng@gwu.edu.*
- *L. Ma and K. Xing are with the Department of Computer Science, The George Washington University, 801 22nd Street NW, Phillips Hall, Suite 712, Washington, DC 20052. E-mail: {lrma, kaix}@gwu.edu.*

Manuscript received 29 June 2006; revised 20 June 2007; accepted 17 Oct. 2007; published online 22 Oct. 2007.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0180-0606.
Digital Object Identifier no. 10.1109/TMC.2007.70775.

slight increase in storage and computation overhead. SBK and iSBK are truly in situ schemes for bootstrapping keys in sensor networks that place no special requirement on regular sensors.

We conduct both theoretic analysis and simulation study to evaluate the performance of SBK. Compared to the majority of existing key predistribution schemes [7], [8], [10], [13], [16], [18], [29], SBK and iSBK have the following characteristics:

- SBK/iSBK is a localized framework that requires no a priori knowledge. This feature is particularly attractive, since in many applications, sensors are dropped from aircraft, and the topology is unpredictable.
- SBK/iSBK has high scalability in network size, the key-sharing probability is high too, each sensor has low storage overhead, and SBK/iSBK has perfect resilience against node capture attacks. To the best of our knowledge, SBK and iSBK are the only ones that achieve these four objectives simultaneously.

The remainder of this paper is organized as follows: Major key predistribution schemes are summarized in Section 2. Preliminaries, models, and assumptions are sketched in Section 3. SBK, the self-configuring framework for bootstrapping keys in sensor networks, is proposed in Section 4 and is evaluated in Section 5. An improved version (that is, iSBK) is elaborated in Section 6. The simulation studies for both schemes are reported in Section 7. Finally, we conclude our paper in Section 8.

2 Related Work: Key Predistribution

In this section, the majority of related work on key predistribution is summarized and compared. We refer the readers to [5] for a more comprehensive literature survey.

The basic *random-keys scheme* is proposed by Eschenauer and Gligor [13], in which a large key pool $\mathcal{K}$ is computed offline, and each sensor picks m keys randomly from $\mathcal{K}$ without replacement before deployment. Two sensors can establish a shared key, as long as they have at least one key in common. To enhance the security of the basic scheme against small-scale attacks, Chan et al. [7] propose the q-composite keys scheme, in which $q > 1$ number of common keys are required for two nodes to establish a shared key. This scheme performs worse in resilience when the number of compromised sensors is large.

In these two schemes [7], [13], increasing the number of compromised sensors increases the percentage of compromised links shared by uncompromised sensors. To overcome this problem, in the same work, Chan et al. [7] propose boosting up a unique key for each link through multipath enhancement. For the same purpose, Zhu et al. [30] propose utilizing multiple logic paths. To improve the efficiency of key discovery in [7] and [13], which is realized by exchanging the identifiers of the stored keys or by a challenge-response procedure, Zhu et al. [30] propose an approach based on the pseudorandom key generator seeded by the node ID. Each sensor computes the key identifiers and preloads the corresponding keys based on its unique ID. Two sensors can determine whether they have a common key based on their IDs only. Note that this procedure does not improve the security of the key discovery procedure, since an attacker can still figure out the key identifiers as long as the algorithm is available. Furthermore, a smart attacker can easily beat the pseudorandom key generator to compromise the network faster [9]. Actually, for smart attackers, challenge response is an effective way for key discovery, but it is too computationally intensive. Di Pietro et al. [9] propose a pseudorandom key predeployment scheme that supports a key discovery procedure that is as efficient as the pseudorandom key generator [30] and is as secure as challenge response.

To improve the resilience of the random keys scheme against node capture attacks, *random-pairwise-keys schemes* have been proposed [7], [8], in which a key is shared by two sensors only. These schemes have good resilience against node capture attacks, since the compromise of a sensor only affects the links incident to that sensor. The difference between [7] and [8] is that the sensors in [7] are paired based on their IDs, whereas in [8], they are on virtual grid locations. Similar to the random-keys schemes, random-pairwise-keys schemes do not scale well to large sensor networks. Neither do they have good key-sharing probability due to the conflict between the high keying storage redundancy and the memory constraint.

To improve the scalability of the random-keys schemes, two *random-key-spaces schemes* [10], [16] have been proposed independently in the *10th ACM Conference on Computer and Communications Security (CCS 2003)*. These two are similar in nature, except that they apply different key space models, which will be summarized in Section 3.1. Each sensor preloads several keying shares, with each belonging to one key space. Two sensors can establish a shared key if they have keying information from the same key space. Liu and Ning [16] also propose assigning one key space to each row or each column of a virtual grid. A sensor residing at a grid point receives keying information from exactly two key spaces. This realization involves a larger number of key spaces. Note that these random key space schemes also improve resilience and key-sharing probability because more key spaces are available and two sensors compute a unique key within one key space for their shared links.

Compared to the work mentioned above, *group-based schemes* [11], [18], [29] have the best performance in scalability, key-sharing probability, storage, and resilience due to the relatively least randomness involved in these key predistribution schemes. Du et al.'s scheme [11] is the first to apply the group concept, in which sensors are grouped before deployment, and each group is dropped at one deployment point. Correspondingly, a large key pool $\mathcal{K}$ is divided into subkey spaces, 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 a higher chance of establishing a pairwise key directly. However, the strong assumption on the deployment knowledge (static deployment point) renders it impractical for many applications. It is interesting to observe that two similar works [18], [29] have been proposed in the *Fourth ACM Workshop on Wireless Security (WiSe 2005)* independently. In [18], sensors are equally partitioned based on IDs into disjoint *deployment groups* and disjoint *cross groups*. Each sensor resides in exactly one deployment group and one cross group. Sensors within the same group can establish shared keys based on any of the key establishment schemes mentioned above [3], [4], [7], [8],

[13]. In [29], the deployment groups and cross groups are defined differently, and each sensor may reside in more than two groups. Note that these two schemes inherit many nice features of [11], but they release the strong topology assumption adopted in [11]. A major drawback of these schemes is the high communication overhead when path keys are sought to establish shared keys between neighboring sensors.

Even with these good efforts, the shared key establishment problem still has not been completely solved yet. As claimed in [12] and [24], the performance is still constrained by the conflict between the desired probability to construct shared keys for communicating parties and the resilience against node capture attacks under a given capacity for keying information storage in each sensor. Researchers have been actively working toward this to minimize the randomness [17], [14] in the key predistribution schemes. Due to space limitations, we could not cover all of them thoroughly. Interested readers are referred to a recent survey [5] and the references therein.

Architectures consisting of base stations for key management have been considered in [22] and [31], which rely on base stations to establish and update different types of keys. In [6], Carman et al. apply various key management schemes on different hardware platforms and evaluate their performance in terms of energy consumption, etc. Authentication in sensor networks has been considered in [22], [26], [31], etc.

Our work is radically different from all those mentioned above in that SBK provides a pure in situ framework for bootstrapping keys in sensor networks. There is no predistribution of any keying information. Sensors differentiate their roles as either service nodes or worker nodes after deployment through a bootstrapping procedure, and service nodes generate and distribute keying information to facilitate the bootstrapping of pairwise keys between neighboring worker sensors. The idea of SBK is fundamentally different from all key predistribution schemes. Because the randomness inherent to all predistribution schemes has been completely removed, SBK achieves high scalability in network size, high key-sharing probability, and low storage overhead. Furthermore, because SBK explores the λ-collusion-resistant property of the underlying key space models, it has perfect resilience against node capture attacks.

3 Preliminaries, Models, and Assumptions

3.1 Key Space Models

The two key space models for establishing pairwise keys—one is polynomial based [4], and the other is matrix based [3]—have been tailored for sensor networks in [16] and [10], respectively. These two models are similar in nature.

The polynomial-based key space utilizes a bivariate λ-degree polynomial $f(x,y) = f(y,x) = \sum_{i,j=0}^{\lambda} a_{ij}x^j y^j$ over a finite field F_q, where q is a large prime number.[4] By plugging in the ID of a sensor, we obtain the keying information (called a *polynomial share*) allocated to that sensor. For example, sensor i receives $f(i,y)$. Therefore, two sensors can compute a shared key from their keying information as $f(x,y) = f(y,x)$. For the generation of a polynomial-based key space $f(x,y)$, we refer the readers to [4].

The matrix-based key space utilizes a $(\lambda+1)\times(\lambda+1)$ public matrix[5] G and a $(\lambda+1)\times(\lambda+1)$ private matrix D over a finite field $GF(q)$, where q is a prime that is large enough to accommodate a cryptographic key. We require D to be symmetric. Let $A = (D \cdot G)^T$. If D is symmetric, $A \cdot G$ is symmetric too. If we let $K = A \cdot G$, we have $k_{ij} = k_{ji}$, where k_{ij} is the element at the ith row and the jth column of K, $i,j = 1,2,\cdots,\lambda+1$. Therefore, if a sensor knows a row of A, say, row i, and a column of G, say, column j, then the sensor can compute k_{ij}. Based on this observation, we can allocate to sensor i a keying share containing the ith row of A and the ith column of G such that two sensors i and j can compute their shared key k_{ij} by exchanging the columns of G in their keying information. We call (D,G) a matrix-based key space, whose generation has been well documented in [3] and further in [10].

Both key spaces are λ-collusion resistant [3], [4]. In other words, as long as not more than λ sensors receiving keying information from the same key space release their stored keying shares to an attacker, the key space remains perfectly shared. Note that it is interesting to observe that the storage space required by a keying share from either key space at a sensor can be very close ($(\lambda+1)\cdot\log q$ for the polynomial-based key space [4] and $(\lambda+2)\cdot\log q$ for the matrix-based key space [3]) for the same λ, as long as the public matrix G is carefully designed. For example, Du et al. [10] propose employing a Vandermonde matrix over $GF(q)$ for G such that a keying share contains one row of A and the seed element of the corresponding column in G. However, the column of G in a keying share must be restored when needed, resulting in $(\lambda-1)$ modular multiplications.

Note that our SBK framework for bootstrapping keys works with both key space models. Service sensors need to generate a key space and then distribute to each worker sensor a keying share. Two worker sensors can establish a shared key, as long as they have keying information from the same key space. Note that, for a polynomial-based key space, two sensors need to exchange their IDs, whereas for a matrix-based key space, they need to exchange the columns (or the seeds of the corresponding columns) of G in their keying shares. In this paper, we will elaborate SBK in a higher level without specifying any of the key space models since both work fine for us.

3.2 Rabin's Public Cryptosystem

We need Rabin's public cryptosystem [23] as a cryptographic primitive to establish a computationally asymmetric secure channel through which keying information 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,

4. The number q must be large enough to accommodate a cryptographic key.

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

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). For details on Rabin's public cryptosystem, we refer the readers to [21] or [23].

3.3 Network Model and Security Assumptions

We target a large-scale sensor network with homogeneous sensors dropped over the deployment region through vehicles such as aircraft. Therefore, no topology information is available before deployment. Sensors are preloaded with several system parameters and differentiate their roles as either worker nodes or service nodes after deployment. Worker sensors are in charge of sensing and reporting data and thus are expected to operate for a long time. Service sensors take care of key space generation and keying information dissemination to assist in bootstrapping pairwise keys among worker sensors. They may die early due to depleted energy resulting from high workload in the bootstrapping procedure. In this sense, they are sacrifices. Nevertheless, we assume that service sensors are able to survive the bootstrapping procedure. Note that all sensors are motelike and have limited storage for data and code. For example, the Mica2 mote [1] has 128-Kbyte program flash memory, 512-Kbyte measurement flash that can support more than 100,000 measurements, and 4-Kbyte configuration Eeprom. We assume that the measurement flash can be used to hold a key space within a service node during the configuration procedure.

Since all sensors are homogeneous, their transmission ranges are assumed to be the same. Therefore, the induced network topology is a disk graph. Two sensors are neighbors if they are located within each other's transmission range. They are called one-hop or immediate neighbors. We further assume that there exists a reliable MAC protocol such as [28] to schedule the local broadcastings in SBK for collision avoidance.

In our consideration, sensors are not tamper resistant. The compromise or capture of a sensor releases all its security information to the attacker. However, as argued in [2] and [31], 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;[6] otherwise, the whole network can be easily taken over by the opponent. Therefore, we can safely assume that, during the bootstrapping procedure, attackers can only passively eavesdrop some of the messages. Note that a *global passive adversary* that can monitor all communications everywhere in the deployment region at all times is a too-strong security model [2]. We employ a key k_0 shared by all sensors preloaded before deployment such that all messages exchanged in the node configuration procedure can be protected by a popular symmetric cryptosystem such as AES or Triple-DES. This key will be used by worker sensors and service sensors to mutually authenticate each other against *sensor injection attacks*.[7] Note that k_0 should be strong enough such that it is almost impossible for an adversary to recover it when obtaining a number of ciphertexts before the bootstrapping procedure is complete. Let T_{k_0} be the minimum time needed to recover k_0. It is reasonable to assume that $T_b << \min\{T_{survival}, T_{k_0}\}$, where T_b is the maximum time needed by the bootstrapping procedure. For example, we can choose k_0 to be 128 bits when AES is applied as the symmetric cryptosystem.

We assume that sensors are coarsely time synchronized by adopting techniques such as those proposed in [15] and [25] before deployment such that they can start the bootstrapping procedure roughly simultaneously. We also assume that all sensors are roughly deployed at the same time. In other words, the total deployment time is much smaller compared to $T_{survival}$ and T_{k_0}. Furthermore, we assume that routing is out of the scope of this paper. Note that these assumptions are practical and have been adopted by many of the related works (for example, [2] and [31]).

4 SBK: The Self-Configuring Framework for Bootstrapping Keys

In this section, we elaborate SBK, a self-configuring framework for bootstrapping keys in large-scale sensor networks. SBK consists of three phases. In the first phase, a probability-based approach is applied to elect service sensors. Each service sensor constructs a key space for later use. In the second phase, each worker sensor associates itself with service sensors in its neighborhood and acquires keying information from each of them through a computationally asymmetric secure channel. In the third phase, two worker sensors derive a shared key if they are associated with the same key space.

Note that all messages in SBK are authenticated via k_0 through a symmetric algorithm such as AES or Triple-DES. We will not emphasize this in the following elaboration. In addition, note that *after the bootstrapping procedure for key distribution terminates, all sensors erase their key space information and k_0 for security enhancement*.

4.1 Service Node Determination and Key Space Computation

In SBK, the selection and configuration of service sensors are controlled automatically by the following preloaded bootstrapping program:

Algorithm 1: Node Self-configuration.

1: **function** $R_l = \textbf{NodeConfig}(\lambda, H, P_s, T_s, t, p, q)$ ▷ R_l is the selected role
2: $eligible \leftarrow true$
3: **while** $eligible\ and\ (t > 0)$ **do** ▷ Election and configuration
4: $TTL \leftarrow T_s$
5: $success \leftarrow electServiceNode(P_s)$ ▷ Elect itself as a service node with probability P_s
6: **if** $success$ then ▷ Elected as a service node
7: $R_l \leftarrow ServiceNode$
8: $\mathcal{K} \leftarrow getKeySpace()$ ▷ construct a key space $\mathcal{K}$
9: $KeyingInfoDistr(\lambda, H, \mathcal{K}, p, q)$ ▷ Algorithm 2: keying information distribution
10: $eligible \leftarrow false$
11: **else**

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

7. Adversaries may inject service sensors, worker sensors, or both immediately after legitimate sensors are deployed. Without protection, injected sensors can participate in the bootstrapping procedure to obtain pairwise keys shared by worker sensors in order to retrieve secure information in the future.

12: **if** $haveServiceNode((H\text{-}1)\text{-}hop)$ **then** ▷ Select to be a worker node if having service nodes within H-1 hops
13: $R_l \leftarrow WorkerNode$
14: $eligible \leftarrow false$
15: **end if**
16: **end if**
17: $t \leftarrow t - 1$
18: **while** $TTL > 0$ **do**
19: $elapse(TTL)$
20: **end while**
21: **end while**
22: **if** $undefined(R_l)$ **then** ▷ Select to be a worker node if failing in all elections
23: $R_l \leftarrow WorkerNode$
24: **end if**
25: **return** R_l
26: **end function**

Several preconfigured system parameters, listed in Table 1, are uploaded to each sensor before deployment. k_0, as explained in Section 3.3, is employed to secure the bootstrapping procedure against eavesdropping attacks and sensor injection attacks. λ, which is the security parameter, determines the maximum number of worker nodes to be served by a service sensor. H, which is 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 self election is denoted by P_s. The initial values of H and P_s are determined by λ according to the following criteria:

$$D_H \leq \lambda, \tag{1}$$

$$P_s = 1/\lambda, \tag{2}$$

where D_H is the average number of neighbors within an H hop distance in the network. Note that D_H can be estimated based on the area of the network field, the number of sensors to be deployed, and the nominal transmission range of a sensor node. The duration time per round for service node election is denoted by T_s, whereas the total time needed for key bootstrapping is T_{sbk}. For simplicity, we assume that $T_{sbk} = t \cdot T_s$, where t is the total number of rounds for service node determination. Note that T_s should be long enough to complete one round. In reality, T_s is a function of the sensor's computation power and network characteristics (sparsity, regularity, etc.). We will study the practical setting of T_s based on real sensor parameters in our future research.

In addition, before deployment, each sensor randomly picks up two primes, p and q, from a pool of large primes without replacement.[8] The prime pool is precomputed by high-performance facilities, which is out of the scope of this paper. Primes p and q will be used to form the private key such that Rabin's public cryptosystem [23] can be applied to establish a secure channel for keying information dissemination in the second phase (see Section 4.2).

Right after deployment, a sensor bootstraps and elects itself as a service node with the probability P_s. Whenever a node becomes a service sensor, the bootstrapping program generates a key space consisting of λ keying shares. The service node election process is repeated every T_s time for t rounds. At the beginning of each round, a nonservice sensor that does not have any service node within $H-1$ hops chooses to become a service node with probability P_s. If a sensor succeeds in the self election, it sets up a key space, announces its status to H-hop neighbors after a random delay, and then enters the next phase for keying information dissemination. 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 Section 4.2), the sensor becomes a worker node, erases its primes, and enters the next phase for service sensor association and keying information acquisition. Note that the reception of a service node announcement also suppresses sensors who have self-elected as service nodes but have not broadcast their decisions to broadcast their status. If no service node within $H-1$ hops is detected in the current round, the sensor participates in the next round. The details of the whole node self-configuration procedure are given in Algorithm 1.

8. More primes may be picked up if needed.

TABLE 1
Preloaded System Parameters

k_0	The shared key by all sensors to secure the bootstrap procedure
λ	Security parameter. Also specifies the maximum number of nodes to be served by a service sensor
H	The forwarding bound of the key space advertisement
P_s	The probability of service node election
p, q	Two primes to be used as the private key of Rabin's cryptosystem
T_s	The time of one round for service node election
T_{sbk}	The total time for key bootstrapping

4.2 Service Node Association and Keying Information Acquisition

Once a service sensor finishes the key space construction, it broadcasts a beacon message, notifying others of its existence after a random delay. A worker node receiving the beacon will acquire keying information from the service sensor through a secure channel established based on Rabin's cryptosystem between the two nodes. As illustrated in Fig. 1, the service node association and keying information acquisition is composed of three steps.

4.2.1 Key Space Advertisement

A service node announces its existence through beacon broadcasting when its key space is ready. As illustrated in Fig. 2, the beacon message includes: 1) a unique key space ID, 2) the public key n, where $n = p \times q$ and (p, q) is the corresponding private key preloaded before deployment, and 3) a TTL value that is initialized to be the forwarding bound H. Nodes receiving the message first subtract 1 from TTL and then forward it if the new TTL value is greater 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 keying share after it receives the key space existence notification (see Section 4.2.2).

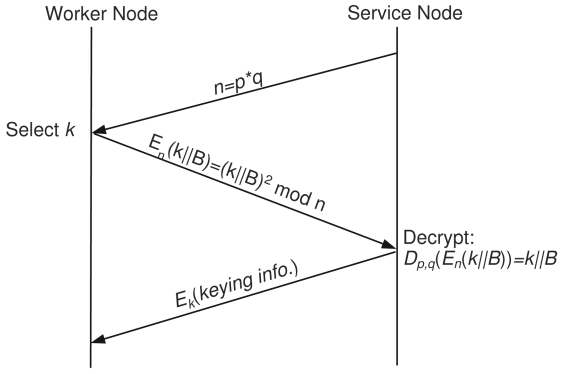


Fig. 1. Service sensor association. A worker node associates itself to a service sensor to obtain the keying information through a secure channel established based on Rabin's public cryptosystem.

As stated in the first 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. 2. If a sensor receives any beacon with $TTL \geq 1$, then at least one service node exists within a $(H-1)$-hop distance. In this case, the node turns out to be a worker sensor and terminates its role determination procedure. Otherwise, the sensor intends to become a service node with probability P_s in the next round.

4.2.2 Secure Channel Establishment

Any worker node requesting the keying information from a service node needs to establish a secure channel to the associated service node. Recall that we leverage Rabin's public key cryptosystem [23] 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.[9] Now, k is known to both the worker sensor and the service sensor and can be used as the secret key of a secure channel for the keying information dissemination in the next step.

The security of Rabin's cryptosystem is based on the factorization of large numbers; thus, it is comparable to that of RSA. Its encryption operation is extremely fast, which is several hundreds of times faster than RSA. However, its decryption time is comparable to RSA. Thus, the establishment of the secure channel shifts a large amount of the computation overhead to service nodes. Note that worker sensors are expected to operate for years, whereas service nodes can die after their duty is complete. In this aspect, service nodes work as sacrifices to extend the network lifetime.

4.2.3 Keying Information Acquisition

After a shared key k is established between a worker node and a service sensor, the service sensor allocates to the worker node a keying share from its key space. Note that a service sensor can only assign keying information upon request to at most λ worker nodes. The keying information, encrypted with k based on any popular symmetric encryption algorithm (AES, DES, etc.), is transmitted to the requesting worker node securely. The behavior of a service node for keying information distribution is summarized in Algorithm 2. Any two worker nodes receiving keying information from the same service node can compute a shared key for secure data exchange in the future.

9. Note that B is used for ambiguity resolution in Rabin's decryption. Therefore, it is transmitted as a plaintext.

id	*n*	*TTL*

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

Fig. 2. The message format of key space advertisement.

Algorithm 2: Keying Information Distribution Algorithm.

1: **function** $\Sigma = \mathbf{KeyingInfoDistr}(I, \lambda, H, P_r^k, \mathcal{K}, p, q)$ ▷ I is the service node ID, Σ is the set containing the keying information that has been allocated
2: $\Sigma \leftarrow \phi$
3: $n \leftarrow p \cdot q$
4: Broadcast (I, n) within H-hop neighborhood ▷ Key space advertisement
5: **while** $|\Sigma| < \lambda$ **do** ▷ Loop for qualified requests
6: **if** $receive(request, E_n(k\|B)\|B)$ **then** ▷ Secure key exchange
7: $k \leftarrow D_{p,q}(E_n(k\|B))$
8: $KeyingInfo \leftarrow$ an unused keyshare
9: $\Sigma \leftarrow \Sigma \cup \{KeyingInfo\}$
10: $send(E_k(KeyingInfo))$
11: **end if**
12: **end while**
13: **return** Σ
14: **end function**

4.3 Shared Key Derivation

Two neighboring nodes sharing at least one key space (having obtained keying information from at least one common service sensor) can establish a shared key accordingly. We refer the readers to Section 3.1 for details on how a shared key can be computed based on the underlying key space model. Note that this procedure involves the exchange of either node ID if a polynomial-based key space model is utilized [4] or columns (seeds) of the public matrix if a matrix-based key space model is utilized [3]. To further improve security, nonces can be introduced against replay attacks.

5 Evaluation of the SBK Scheme: Analytical Results

We consider the following metrics when analyzing SBK: *the key-sharing probability*, *the memory overhead at each sensor for keying information storage*, and *the resilience against node capture attacks*. We also briefly discuss the *computation and communication overheads* of SBK in this section. Note that we will consider worker sensors only. We expect that SBK yields a low memory overhead on worker sensors while achieving a high security against node capture attacks and a high probability in establishing a shared key between two neighboring nodes.

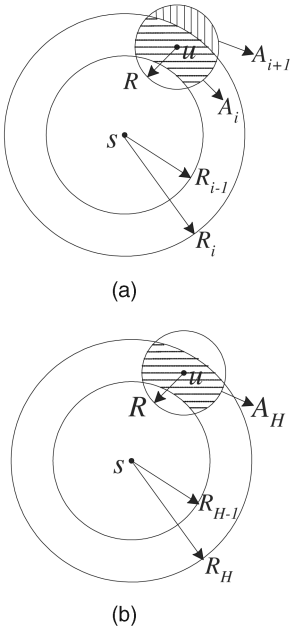


Fig. 3. Two neighboring nodes belong to the same key space. (a) Case 1: v is in area A_i or A_{i+1}. (b) Case 2: v is in area A_H.

5.1 Key Sharing Probability

Let p_k be the probability that two neighboring worker sensors are able to establish a shared key. To study p_k, we need the probability that two neighboring sensors u and v obtain keying information from the same service node s; that is, u and v are both within H hops from s, where H is the forwarding bound for key space existence notification. Since u and v are immediate neighbors, only two possible cases exist:

- *Case 1*. u is the ith-hop neighbor of s and v is the ith-hop or $(i+1)$th-hop neighbor of s, where $i = 1, 2, \ldots, H-1$.
- *Case 2*. u and v are both the 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 [19], which computes an ER R_h to enclose all the t-hop $(t \leq h)$ neighbors within a disk region. As validated in [19], the ER model is an effective approximation when the network is uniformly and densely deployed.

From the ER model, all ith-hop neighbors of s reside in the nonoverlapping area of the two disks defined by R_{i-1} and R_i, as indicated in Fig. 3. Therefore, for case 1, v is either in the area A_i or in the neighbor area A_{i+1}, as shown in Fig. 3a. For case 2, v can only be in the area A_H, as shown in Fig. 3b.

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 neighboring nodes u and v share the key space of the service node s can thus be estimated as

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

where d_h is the number of nodes that are h hops away from s, K_s is the key space provided by service node s, $dist(u,v)$ denotes the distance between u and v, and R is the nominal node transmission range. Hence, the probability p_k that any two neighboring nodes share at least one common key space is

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

where N_s^i is the number of service nodes generated in the ith round computed by (8) and $t = T_{sbk}/T_s$ is the total number of rounds for service node determination.

5.2 Storage Overhead

For simplicity, we count the memory overhead of our protocol as the expected number of key shares (either polynomial shares [16] or matrix shares [10]) that each worker sensor receives from the surrounding service sensors. In SBK, each worker sensor can obtain at most one piece of keying information from a service node. Aside from that, each service node can serve at most λ worker sensors. Therefore, the upper bound of the average number of key shares stored in each worker node can be estimated based on the number of service nodes generated during the bootstrap procedure.

Assume that N sensors are uniformly distributed in the network. Each sensor is preconfigured with the following parameters: λ, H, P_s, T_s, and T_{sbk}, as defined in Table 1. 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 given 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 third round if and only if all of its neighbors within $(H-1)$ hops fail in the first two rounds of elections. Therefore,

$$N_s^3 = (N - N_s^1 - N_s^2) \cdot ((1 - P_s)^2)^{D_{H-1}} \cdot P_s. \quad (7)$$

It follows that the number of service nodes generated in the ith round election is

$$N_s^i = \left(N - \sum_{j=1}^{i-1} N_s^j\right) \cdot \left((1 - P_s)^{(i-1)}\right)^{D_{H-1}} \cdot P_s. \quad (8)$$

Altogether, there are $t = T_{sbk}/T_s$ rounds of service node election. Hence, the average number of keys stored within each worker sensor can be estimated by[10]

$$\tau \approx \lambda \times \frac{\sum_{i=1}^{t} N_s^i}{N - \sum_{i=1}^{t} N_s^i}. \quad (9)$$

10. The actual average is expected to be smaller than τ.

5.3 Resilience

Recall that, in our SBK scheme, a service node can distribute keying information to at most λ worker sensors in its neighborhood and then delete all the key space information, and our key spaces are λ-collusion resistant (see Section 3.1), so it is impossible for an attacker to compromise any key space. Furthermore, 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], [13], 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 larger number of service nodes to be generated in SBK since each worker node should have at least one key to maintain the desired key-sharing probability. Actually, this represents a trade-off between security and cost. In many cases, it is desirable to sacrifice a small fraction of low-cost (service) sensors to achieve a higher level of security. We will further study this by simulation in Section 7.

5.4 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 *keying information acquisition*, and calculating the pairwise keys shared with its neighbors in *shared key derivation*. The first involves one modular squaring, whereas the second requires a symmetric decryption operation. On the average, a worker node completes τ (see (9)) such operations.

In general, given the keying information, computing a shared key with one neighbor takes $(\lambda + 1)$ modular multiplications for both key space models. This must be repeated $d \times p_k$ times per sensor on the 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 keying information, each worker sensor needs $(\lambda - 1) \times \tau$ more modular operations for recovering the complete matrix share.

The communication overhead of the worker sensors in SBK results from requesting keying information from service nodes and relaying messages for others. Each worker sensor also needs to exchange information with its neighbors for shared key derivation. 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 at least one key space.

6 iSBK: The Improved SBK Scheme

For security purposes, we expect that the time for the key bootstrapping will not last too long; that is, we expect that most of the worker nodes can obtain keying information from service nodes and establish secure communications 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 a high key-sharing probability between neighboring sensors while minimizing the whole configuration time T_{sbk}. According to the theoretical analysis in Section 5, the key-sharing probability is dependent on t, where $t = T_{sbk}/T_s$, and the number of service nodes N_s^i generated in each round i for a given network (see (3) and (4)). As indicated by (8), 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 desired key-sharing probability. Therefore, node configuration may take a relatively long time.

To speed up this bootstrapping procedure, we propose an improved scheme, that is, iSBK, which can achieve higher key-sharing probability in a shorter time compared with the basic scheme. Similar to SBK, iSBK also contains three phases: *service sensor determination and key space construction*, *service sensor association and keying information acquisition*, and *shared key derivation*. The differences reside only in the first phase.

6.1 Sensor Determination and Key Space Construction

We modify the initial values for the forwarding bound of key space advertisement H and the probability of service node election P_s according to the following criteria:

$$D_H \leq \lambda, \tag{10}$$

$$P_s = 1/D_{H-1}, \tag{11}$$

where D_h is the expected number of nodes within an h-hop neighborhood, as defined before. Furthermore, the possibility of a node being 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 will compare the percentage of service nodes to be generated in a network regulated with the basic SBK and the iSBK schemes, respectively, through simulation study. The results indicate that iSBK can achieve a better performance with a small increase in the number of service nodes involved.

7 Evaluation on SBK and iSBK: Simulation Results

In this section, we study the performance of both the basic SBK and the iSBK schemes through simulation in terms of key-sharing probability and storage overhead (with only worker sensors considered). We also consider the percentage of service nodes generated, which indicates the cost of our in situ key establishment scheme.

Note that the security of SBK has been studied in Section 5.3. Due to the λ-collusion resistant property of the underlying key space model, SBK exhibits a perfect

resilience against node capture attacks by allowing at most λ worker sensors within the same key space. iSBK follows the same regulation and possesses the same nice property. Thus, after bootstrapping keys with the proposed SBK (iSBK) scheme, no matter how many nodes are compromised, the shared keys of the remaining sensors are still kept intact, and the network can still function well in secure communication.[11]

We measure the performance with the following metrics in our simulation study:

- *Key-sharing probability* p_k. To study the probability that two immediate neighbors share at least one common key space, we count the ratio of the number of neighboring pairs sharing at least one common service node to the total number of neighboring pairs as an approximation.
- *Keying information storage* τ. We count on τ (9), the average number of keying information that a sensor obtains, to assess the storage overhead of a worker sensor in SBK.
- *Percentage of service sensors* N_s/N. Since service nodes intend to be sacrifices, fewer of them are desired. As a metric, we use the percentage of service nodes generated by SBK among all the nodes deployed in the network.

In both the basic SBK and the iSBK schemes, three system parameters affect the performance, i.e., the node density of the network, the maximum number of worker nodes λ to be served by one service node, and the number of rounds for service node election $(t = T_{sbk}/T_s)$. In the following, we design two experiments to study the relationship between these factors and the performance of our schemes. The results are expected to provide guidelines when our schemes are applied to real sensor network deployment. In both experiments, we consider a sensor network deployed over a field of $1{,}000 \times 1{,}000$. A number of sensors are uniformly distributed,[12] and each node is capable of a fixed transmission range of 40. The simulation results are averaged over 50 runs. We also plot the analytical results (from Section 5) for comparison.

In the first experiment, we deploy 2,000 sensors in the region. Fig. 4 shows the results obtained after the *first*, *third*, and *fifth* rounds of service node election. We notice that iSBK achieves a higher key-sharing probability within a much shorter time than the basic SBK scheme. As illustrated in Fig. 4a, $t = 3$ is enough for iSBK to achieve a p_k close to 1. With the same amount of time, at most 80 percent of the neighboring pairs can establish secure communication using the basic SBK. We also notice that, compared to the basic SBK, iSBK generates more service nodes, and therefore, worker sensors carry more key spaces, as indicated in Figs. 4b and 4c. Nonetheless, in iSBK, each worker node still consumes only a small amount of memory $(\tau \approx 2)$ to achieve high key-sharing probability. Note that both schemes achieve a desirable level of p_k at the expense of a very small memory overhead

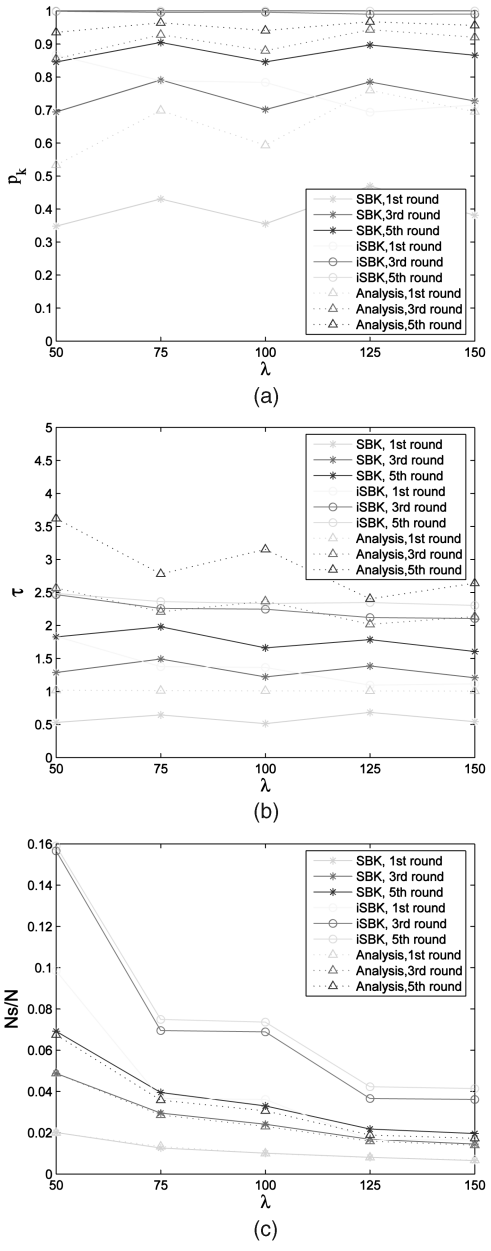


Fig. 4. Test 1: performance of the basic and the iSBK schemes in the *first*, *third*, and *fifth* rounds, with $N = 2{,}000$. (a) p_k: the key-sharing probability. (b) τ: the keying information storage. (c) N_s/N: the percentage of service nodes.

in worker nodes. For example, almost all worker nodes can establish secure communication with their neighbors using iSBK, with each worker sensor storing about two keying shares when $t = 3$. Even with the basic SBK, p_k is above 90 percent, with each worker node carrying less than two key spaces when $t = 5$. In summary, both SBK and iSBK can achieve a high key-sharing probability between neighboring sensors 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 memory overhead in worker nodes. The comparison results confirm that the theoretical analysis results derived in Section 5 are the upper bound of the simulation ones.

11. Key spaces are unique with very high probability since they are independently generated.

12. Although only the uniform distribution is studied in the simulation, both SBK and iSBK can be applied to any network topology.

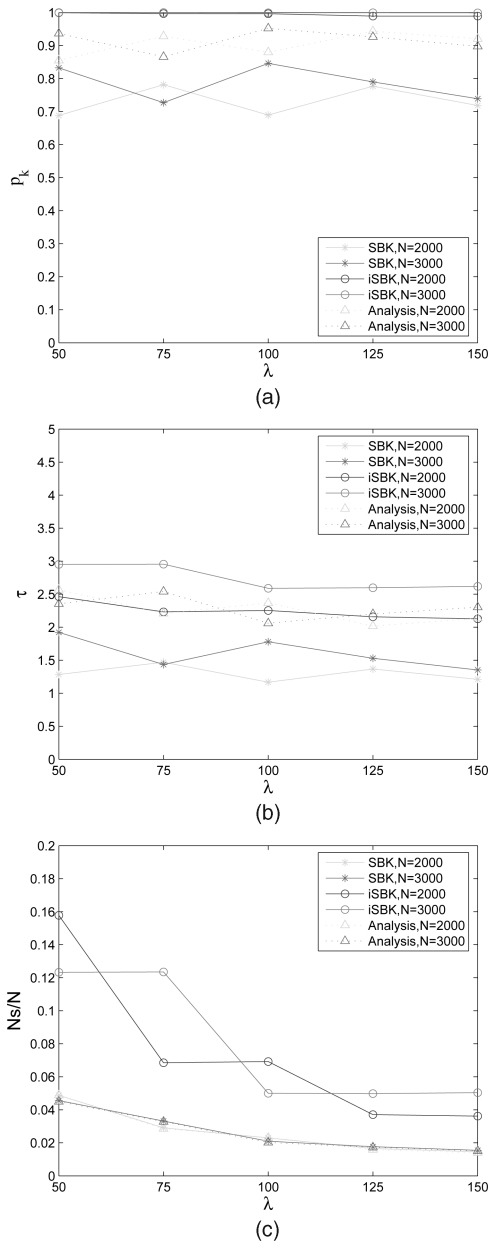


Fig. 5. Test 2: performance of the basic and the iSBK schemes, $t = 3$, with different network densities. (a) p_k: the key-sharing probability. (b) τ: the keying information storage. (c) N_s/N: the percentage of service nodes.

In the second experiment, we study the performance of SBK and iSBK under different network densities, with t fixed to 3. We deploy $N = 2{,}000$ or $N = 3{,}000$ nodes in the network area. As illustrated in Fig. 5, iSBK still outperforms the basic one, whereas both achieve similar key-sharing probability under different node densities. We also notice that, under a given λ, a higher node density does not necessarily result in a larger percentage of service nodes nor a heavier memory overhead since each service sensor is expected to serve λ worker nodes in the vicinity.

In Figs. 4c and 5c, SBK exhibits a "smoother" curve, whereas iSBK may generate a similar percentage of service sensors for some different λ values. The reason is that, for iSBK, different values of λ may result in the same H and P_s pair under a given network density according to (10) and (11). Specifically, in our experiments, $\lambda = 75, 100$ lead to the same pair of H and P_s, and so do $\lambda = 125, 150$ when $N = 2{,}000$. In the case of $N = 3{,}000$, $\lambda = 50, 75$ lead to the same pair of H and P_s, and so do $\lambda = 100, 125, 150$.

Furthermore, both experiments also indicate that SBK and iSBK achieve similar key-sharing probability and memory overhead for different λ values. This indicates that our schemes make sensors "self-configure" according to the environments (system parameters, node density, etc.) and elect service nodes when necessary. According to (1), (2), (10), and (11), 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 with 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. 4c and 5c, where the number of service nodes decreases as λ increases.

Note that we have also conducted extensive simulations to compare multiple existing key predistribution schemes [7], [10], [13], [16] with SBK and iSBK in terms of key-sharing probability and storage overhead, and we have achieved results very favorable to SBK and iSBK. For example, with the same parameter settings for λ and τ, for SBK, $p_k > 80$ percent, whereas for [10], $p_k \approx 15$ percent. Due to space limitations, these results have to be reported in a different paper.

8 Conclusion and Future Work

In this paper, we propose SBK, a self-configuring framework for bootstrapping keys in sensor networks. SBK is fundamentally different compared to all existing key predistribution schemes. It is an in situ key establishment framework that achieves high scalability in network size since it is purely localized. It exhibits a perfect resilience against node capture attacks by exploring the λ-collusion resistant property of the underlying key space model. Since SBK does not require predistribution of random-key-space-related information, the randomness inherent to all key predistribution schemes has been completely removed. Therefore, SBK achieves a high key-sharing probability with a 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, key-sharing probability, storage overhead, and resilience.

As our future research, we will study the impact of the settings of the initial system parameters such as T_s, T_{sbk}, P_s, and H on the performance of SBK. We intend to design a new topology-adaptive in situ key establishment scheme whose parameter settings can be adaptively adjusted based on the local network topology. Practical settings of T_s based on the real sensor's computation power will also be explored. We also plan to quantitatively study the resource consumption and lifetime of service sensors. This research plays a significant role in the performance of SBK.

References

[1] *Crossbow MPR400/410/420 MICA2 Mote,* www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf, 2007.

[2] R. Anderson, H. Chan, and A. Perrig, "Key Infection: Smart Trust for Smart Dust," *Proc. 12th IEEE Int'l Conf. Network Protocols (ICNP '04),* 2004.

[3] R. Blom, "An Optimal Class of Symmetric Key Generation Systems," *Proc. Workshop Theory and Application of Cryptographic Techniques (EUROCRYPT '84),* 1984.

[4] C. Blundo, A.D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-Secure Key Distribution for Dynamic Conferences," *Proc. 12th Ann. Int'l Cryptology Conf. (CRYPTO '92),* 1992.

[5] S.A. Camtepe and B. Yener, "Key Distribution Mechanisms for Wireless Sensor Networks: A Survey," Technical Report TR-05-07, Rensselaer Polytechnic Inst., Mar. 2005.

[6] D.W. Carman, P.S. Kruss, and B.J. Matt, "Constraints and Approaches for Distributed Sensor Network Security," Technical Report 00-010, NAI Laboratories, Sept. 2000.

[7] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," *Proc. IEEE Symp. Security and Privacy (SP '03),* 2003.

[8] H. Chan and A. Perrig, "PIKE: Peer Intermediaries for Key Establishment in Sensor Networks," *Proc. IEEE INFOCOM '05,* Mar. 2005.

[9] R. Di Pietro, L.V. Mancini, and A. Mei, "Efficient and Resilient Key Discovery Based on Pseudo-Random Key Pre-Deployment," *Proc. 18th Int'l Parallel and Distributed Processing Symp. (IPDPS '04),* 2004.

[10] W. Du, J. Deng, Y.S. Han, and P.K. Varshney, "A Pairwise Key Predistribution Scheme for Wireless Sensor Networks," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03),* 2003.

[11] W. Du, J. Deng, Y.S. Han, S. Chen, and P. Varshney, "A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge," *Proc. IEEE INFOCOM '04,* Mar. 2004.

[12] W. Du, R. Wang, and P. Ning, "An Efficient Scheme for Authenticating Public Keys in Sensor Networks," *Proc. ACM MobiHoc '05,* 2005.

[13] L. Eschenauer and V.D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," *Proc. Ninth ACM Conf. Computer and Comm. Security (CCS '02),* 2002.

[14] D. Huang, M. Mehta, D. Medhi, and L. Harn, "Location-Aware Key Management Scheme for Wireless Sensor Networks," *Proc. Second ACM Workshop Security in Ad Hoc and Sensor Networks (SASN '04),* 2004.

[15] Q. Li and D. Rus, "Global Clock Synchronization in Sensor Networks," *Proc. IEEE INFOCOM '04,* 2004.

[16] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03),* 2003.

[17] D. Liu and P. Ning, "Location-Based Pairwise Key Establishments for Static Sensor Networks," *Proc. First ACM Workshop Security in Ad Hoc and Sensor Networks (SASN '03),* 2003.

[18] D. Liu, P. Ning, and W. Du, "Group-Based Key Predistribution in Wireless Sensor Networks," *Proc. Fourth ACM Workshop Wireless Security (WiSe '05),* 2005.

[19] L. Ma, W. Jiang, K. Xing, and E.K. Park, "The Effective Radius Model for Multihop Wireless Networks," *Proc. First Int'l Conf. Wireless Algorithms, Systems, and Applications (WASA '06),* 2006.

[20] G. Marsaglia, A. Zaman, and W.W. Tsang, "Towards a Universal Random Number Generator," *Statistics and Probability Letters,* vol. 8, pp. 35-39, 1990.

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

[22] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J.D. Tygar, "SPINS: Security Protocols for Sensor Networks," *Proc. ACM MobiCom '01,* July 2001.

[23] M.O. Rabin, "Digitalized Signatures and Public-Key Functions as Intractable as Factorization," Technical Report MIT/LCS/TR-212, Laboratory for Computer Science, Massachusetts Inst. Technology, 1979.

[24] E. Shi and A. Perrig, "Designing Secure Sensor Networks," *IEEE Wireless Comm.,* vol. 11, no. 6, Dec. 2004.

[25] B. Sundararaman, U. Buy, and A.D. Kshemkalyani, "Clock Synchronization in Wireless Sensor Networks: A Survey," *Ad Hoc Networks,* May 2005.

[26] R. Watro, D. Kong, S.-F. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK: Securing Sensor Networks with Public Key Technology," *Proc. Second ACM Workshop Security in Ad Hoc and Sensor Networks (SASN '04),* Oct. 2004.

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

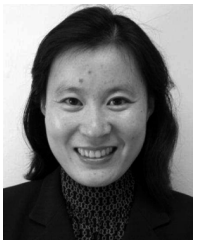
[28] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," *Proc. IEEE INFOCOM '02,* June 2002.

[29] L. Zhou, J. Ni, and C.V. Ravishankar, "Efficient Key Establishment for Group-Based Wireless Sensor Networks," *Proc. Fourth ACM Workshop Wireless Security (WiSe '05),* 2005.

[30] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Establishing Pairwise Keys for Secure Communication in Ad Hoc Networks: A Probabilistic Approach," *Proc. 11th IEEE Int'l Conf. Network Protocols (ICNP '03),* Nov. 2003.

[31] S. Zhu, S. Setia, and S. Jajodia, "Leap: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03),* 2003.

Fang Liu received the DSc degree in computer science from George Washington University in August 2007. She is an assistant professor in the Department of Computer Science, University of Texas, Pan American. Her research interests include wireless and mobile computing, wireless security, and algorithm design and analysis. She is a member of the IEEE.



Xiuzhen (Susan) Cheng received the MS and PhD degrees in computer science from the University of Minnesota, Twin Cities, in 2000 and 2002, respectively. She is currently an assistant professor in the Department of Computer Science, George Washington University. She has served on the editorial boards of several technical journals and in the technical program committees of various professional conferences/workshops. She is the steering committee vice chair of the International Conference on Wireless Algorithms, Systems, and Applications (WASA). She also served as a program cochair of WASA 2006. She was a program director with the US National Science Foundation for six months in 2006. Her research interests include wireless and mobile computing, sensor networking, wireless and mobile security, and algorithm design and analysis. She is a member of the IEEE and the IEEE Computer Society. She received a US National Science Foundation Faculty Early Career Development (CAREER) Award in 2004.

Liran Ma received the BS degree in electrical engineering from Hunan University, Changsha, China, in 1999 and the MS degree in communication and information systems from Northern Jiaotong University, Beijing, in 2003. He is currently working toward the PhD degree in computer science at the George Washington University. His research activity is focused on wireless networks and distributed systems, in particular, wireless LANs, ad hoc networks, sensor networks, and mesh networks. His research interests include algorithm design and analysis, wireless and Internet security, and network protocol design. He is a student member of the IEEE and the IEEE Computer Society.

Kai Xing received the BSc degree in computer science from the University of Science and Technology of China in 2003. He is currently working toward the PhD degree in the Department of Computer Science, George Washington University. His research interests are mobile computing and wireless networks. His current research is focused on multiradio multichannel wireless networks and in-network information assurance in ad hoc and sensor networks. He is a student member of the IEEE.