Chapter 1

CBKE: Chord-based Key Establishment Schemes for Wireless Sensor Networks

Zhijie Jerry Shi, Bing Wang, Fan Zhang
Department of Computer Science and Engineering,
University of Connecticut, Storrs, CT 06269

Because of limited resources at sensor nodes, sensor networks typically adopt symmetric-key algorithms to provide security functions such as protecting communications between nodes. In order to use symmetric-key algorithms, two nodes need to establish a secret session key first. In this chapter, we describe a novel chord-based key establishment (CBKE) protocol that allows any pair of nodes in a sensor network to establish a secret session key. CBKE is a generalized deterministic key establishment scheme that provides great flexibility for balancing memory overhead, reliability, and communication cost. We also analyze the properties of CBKE and explore the performance trade-offs using simulation.

1.1. Introduction

As wireless sensor networks are adopted in more and more applications, the security of sensor networks becomes increasingly important because sensor nodes are deployed in the field and communicate through wireless channels, thus vulnerable to attacks of many kinds. Like in many other systems, security goals in wireless sensor networks are achieved through cryptographic algorithms. Major types of cryptographic algorithms to provide confidentiality include symmetric-key and public-key algorithms. Typically, sensor networks adopt symmetric-key algorithms since public-key algorithms are computationally expensive while sensor nodes have limited computational capability and battery energy. To use symmetric-key algorithms, two sensor nodes need to agree on a secret key first. This process is called *key establishment* (or *key exchange*). While public-key algorithms are used to perform key establishment in systems that have more resources, it is desirable to perform key establishment in sensor networks with symmetric-key algorithms only, for the same reasons described earlier.

In the past several years, key establishment in sensor networks has attracted a lot of attention (see [2] for a survey). In this chapter, we focus on *pairwise* key es-

A preliminary version of this chapter appeared in [1]. This research was supported by National Science Foundation CAREER awards 0644188 and 0746841, and the University of Connecticut Large Grant.

tablishment that allows any two nodes in the network to agree on a secret key, since pairwise communications are the most common cases in sensor networks [3][a]. Existing pairwise key establishment schemes can be broadly classified into the following five major categories: *trusted-server-based schemes, random key pre-distribution schemes, deterministic key pre-distribution schemes, location-based schemes, and in-situ schemes*. Trusted-server-based schemes (e.g., [9]) use a trusted server as an arbiter to establish pairwise keys between two nodes. Random key pre-distribution schemes (e.g., [10, 11]) preload a set of keys chosen from a global key pool into each node. Two neighboring nodes share a common key with a certain probability. Two nodes not presharing a key establish keys by finding a path in the induced key-sharing graph (where two nodes are connected if they share a key). The performance of random key pre-distribution schemes can be improved by using multiple key spaces (e.g., [12, 13]) or by regulating the nodes that share each seed [14]. Deterministic key pre-distribution schemes pre-distribute keys in a deterministic manner. One extreme form of deterministic key pre-distribution is full pairwise key pre-distribution in which every pair of nodes share a unique key and each node has to store as many keys in its memory as the number of nodes in the network. The other extreme is that all nodes preload a single key. PIKE [15] is between these two extremes. It considers the node ID space as a logical grid. Any two nodes in the same row or column preshare a unique key; nodes not presharing a key establish keys through intermediate nodes. Location-based schemes (e.g., [4, 5, 16]) take advantage of location information to improve key establishment. In in-situ schemes [17–19], instead of preloading keys, sensors compute keys after deployment to adapt to the post-deployment network topology.

Several other types of key establishment schemes are proposed recently. For example, Teymorian et al. propose a cellular automata based scheme that allows sensors to establish pairwise keys during any stage of the network operation [20]; Zhang et al. propose a random perturbation-based pairwise key establishment scheme that utilizes multiple perturbed polynomials to establish keys between any pair of nodes directly [3].

In this chapter, we study a novel chord-based key establishment (CBKE) protocol for pairwise key establishment. CBKE is a deterministic key pre-distribution scheme. It is very flexible and achieves a wide range of performance trade-offs among memory overhead, reliability, and communication cost. As we shall see, existing deterministic key distribution schemes, including full pairwise pre-distribution, single network-wide key scheme, and PIKE, are special cases of CBKE. By choosing parameters properly, CBKE can meet various design goals. For example, in networks that have a large number of nodes and require low memory overhead, CBKE can reduce the number of preloaded keys to $\log_2 N + 1$, where N is the number of nodes in a network.

[a]Key establishment in sensor networks can also be group-based (for multicast as in [4–6]) or network-based (for broadcast as in [7, 8]).

The rest of the chapter is organized as follows. Section 1.2 describes the CBKE scheme. We discuss its properties in Section 1.3 and study performance trade-offs in Section 1.4. We then examine the security of CBKE in Section 1.5. After discussing several improvements in Section 1.6, we conclude the chapter in Section 1.7.

1.2. Chord-based Key Establishment Scheme

In this section, we first describe the basic CBKE and then extend it to a generalized scheme.

1.2.1. *Basic CBKE scheme*

Since CBKE employs deterministic key pre-distribution, we start with the key pre-distribution process. We then describe the key establishment process and a mechanism to improve reliability.

1.2.1.1. *Key pre-distribution in basic CBKE*

Consider a sensor network of N nodes. Node i is assigned an ID of i, $i = 0, \ldots, N-1$. For simplicity, we assume $N = 2^m$. So each ID is represented with m bits. Inspired by [21], we consider that these nodes are arranged in a circle in the order of their ID, which increases in a clockwise direction. The *distance* between two nodes is defined as the length of the shorter path along the circle from one node to the other. This shorter path can be either clockwise or counterclockwise. Let $D(i, j)$ denote the distance between nodes i and j. It can be computed as $D(i, j) = \min(i - j, j - i)$[b]. When preloading keys, we let node i preshare a key with nodes $i \pm 2^0, i \pm 2^1, \ldots, i \pm 2^{m-2}, i + 2^{m-1}$. That is, nodes i and j preshare a key if and only if the distance between them is a power of two, i.e., $D(i, j) = 2^x$ for some $x \in \{0, \ldots, m-1\}$. Let $K_{i,j}$ denote the key preshared by nodes i and j. Then $K_{i,j} = K_{j,i}$. On the circle, we connect two nodes with a chord if they preshare a key. During key establishment, a session key is forwarded along the chords from a source to a destination. So we name our scheme a *Chord-based Key Establishment* (CBKE) scheme.

With the above key pre-distribution, each node shares a unique key with $2m - 1 = 2\log_2 N - 1$ nodes. The number of preloaded keys on each node can be reduced to $m + 1$, using the method proposed in [22].

Fig. 1.1 shows an example of key pre-distribution for a network of 16 nodes, in which node 0 preshares keys with nodes 1, 2, 4, 8, 12, 14 and 15. In the figure, node 0 is connected with each of these nodes with a chord.

[b] All computations involving node IDs are modulo N. For simplicity, we do not indicate it explicitly.

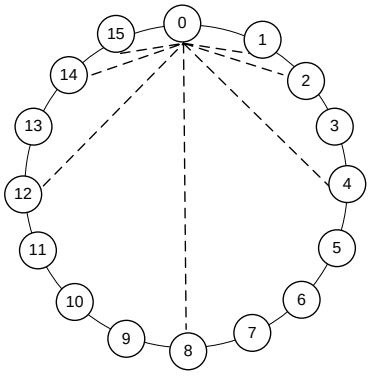

Fig. 1.1. An illustration of key pre-distribution in CBKE ($N = 16$).

1.2.1.2. *Key establishment in basic CBKE*

Suppose that a source node s wants to establish a session key with a destination node d. For this purpose, s first generates a random key r. The key establishment process is basically forwarding the key to d through a secure path (i.e., along chords in the circle); s confirms the success of key establishment if it receives an ACK message from d that is encrypted with r. Let $k = D(s, d)$, k_i denote bit i [c] in the binary representation of k, and $H(k)$ be the *Hamming weight* of k (i.e., the number of 1's in k's binary representation). Since nodes s and d are not the same, $k \neq 0$ and $H(k) > 0$. We now describe the key establishment process, as summarized in Fig. 1.2.

Node s finds a node n_1 that is closer to d and preshares a key with s. In particular, s randomly selects u such that $k_u = 1$ and computes $n_1 = s \pm 2^u$. The operation, $+$ or $-$, depends on whether the shorter path from s to d is clockwise or counterclockwise. If it is clockwise, $+$ is performed. Otherwise, $-$ is performed. Since k has $H(k)$ 1 bits and s may select any of these 1's, s has $H(k)$ different choices for n_1. We refer to these nodes, which can be selected by s to forward the session key, as *descendants* of s. It is clear that n_1 is closer to d than s because $D(n_1, d) = D(s, d) - 2^u < D(s, d)$. After n_1 is selected, s sends (r, d) to n_1. The message is encrypted with K_{s,n_1}, which is the key preshared by s and n_1.

After n_1 receives the encrypted message from s, it recovers r and d with K_{s,n_1}. If n_1 is not the destination node, it selects one of its descendants n_2 and forwards (r, d) to it. If n_2 is not d either, it will repeat the process. Since the session key r gets closer to d after each hop, it will arrive at d when all the bits in the distance k are changed to 0.

Once d receives r, it sends an ACK message to s. The ACK message can be (s, d, ρ) encrypted with r, where ρ is a random value generated by d. If s can retrieve

[c] We adopt the little-endian convention in this chapter. So bit 0 is the least significant bit, i.e., the right most bit.

At the source node s:
Generate a session key r
$k = D(s, d) = (k_{m-1}, \ldots, k_0)_2$
Randomly choose u such that $k_u = 1$
if s is closer to d clockwise
  Compute $n_1 = s + 2^u$
else
  Compute $n_1 = s - 2^u$
Encrypt (r, d) using key K_{s,n_1}
Send the encrypted message to node n_1

At an intermediate node or the destination node n_i:
Decrypt the received message to recover (r, d)
if $(n_i \neq d)$ { // Not the destination
  $k = D(n_i, d) = (k_{m-1}, \ldots, k_0)_2$
  Randomly choose u such that $k_u = 1$
  if n_i is closer to d clockwise
    Compute $n_{i+1} = n_i + 2^u$
  else
    Compute $n_{i+1} = n_i - 2^u$
  Encrypt (r, d) using key $K_{n_i,n_{i+1}}$
  Send the encrypted message to node n_{i+1}
}
else { // At the destination
  Generate a random value ρ
  Encrypt (s, d, ρ) with r
  Send the encrypted (s, d, ρ) to s
}

Fig. 1.2. Basic CBKE scheme: node s establishes a session key with node d.

s and d with r from the ACK message, it assumes that d has received r successfully. Then it can start a secure communication session with d.

We refer to a path along which a session key is forwarded securely from s to d as a *logical path*. On a logical path, a node and its descendant form a *logical hop* since they are not necessarily neighboring nodes in the physical network. Two nodes on a logical hop preshare a key, which is used to protect key forwarding messages. A logical path is only used for key forwarding — the ACK message is sent directly from d to s; after the session key is established, the secure communications between s and d also take a direct path.

We illustrate the basic scheme with an example shown in Fig. 1.3. Suppose $N = 256$ and node 0 wants to send a session key r to node 83. The distance between the two nodes is $k = D(0, 83) = 83 = 01010011_2$. Since $H(k) = 4$, node 0 has four descendants: nodes $64 = 0 + 2^6$, $16 = 0 + 2^4$, $2 = 0 + 2^1$, and $1 = 0 + 2^0$, where the exponents 6, 4, 1, and 0 are the position of 1's in k. Node 0 randomly picks one of the descendants, say node 64, and forwards r and the destination ID

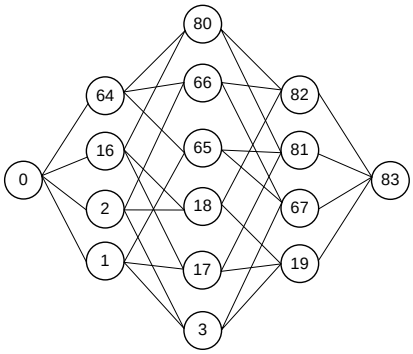

Fig. 1.3. Logical paths from source 0 to destination 83.

83 to it. The message is protected with key $K_{0,64}$. After receiving and decrypting the message, node 64 finds out that it is not the destination. It then forwards the message to one of its descendants (nodes 65, 66, and 80). This process continues until node 83 receives r.

1.2.1.3. *Utilizing multiple logical paths*

In the basic CBKE scheme shown in Fig. 1.2, a session key is forwarded along a single logical path from s to d. A single path, however, does not provide sufficient reliability for key establishment if nodes may fail. As we see in Fig. 1.3, many logical paths can be utilized to forward a key from s to d. Forwarding a session key simultaneously through multiple logical paths improves reliability. The key can reach the destination node even if all but one logical path are broken.

We introduce a parameter l to our basic scheme and allow a node to forward a session key to l descendants simultaneously. In particular, a node n_i forwards the session key to $\min(l, H(D(n_i, d))$ descendants simultaneously. The message to each of the l descendants is protected by the unique key preshared by n_i and its descendants. A descendant may receive the same session key multiple times due to the overlaps among logical paths, but it only forwards the key once.

When $l = 1$, only a single logical path is used to forward the session key from s to d, as illustrated in Fig. 1.2. When not restricting l, i.e., a node forwards the key to all its descendants, CBKE works in the *flooding* mode. There is a clear trade-off in choosing l: a large l improves the reliability of key establishment while increasing communication cost. We will explore the trade-offs in Section 1.4.

We again use the example in Fig. 1.3 to illustrate forwarding along multiple paths. Suppose $l = 2$. Each node forwards the session key to up to two descendants. Source node 0 randomly chooses two nodes from its four descendants, nodes 64, 16, 2, and 1. Suppose nodes 64 and 16 are selected. Node 0 forwards the session key to these two nodes simultaneously. The message to node 64 is encrypted with $K_{0,64}$ and the message to node 16 is encrypted with $K_{0,16}$. After receiving the (encrypted)

session key, node 64 chooses two descendants from nodes 65, 66 and 80. Nodes 16 also chooses two of its descendants randomly. Note that node 80 is a descendant for both nodes 64 and 16. It may receive the session key twice, but it only forwards the key once. Since the session key is forwarded along multiple paths, it can reach the destination node 83 even if some nodes (e.g., node 16) fail.

To reduce communication cost, multiple logical paths can be utilized in a different way. They do not have to be involved in forwarding simultaneously. Instead they can be utilized in series or in a combination of parallel and series. The source node s can keep trying to forward a session key to d with a certain l that incurs less communication cost (e.g. $l = 1$ or $l = 2$) until the key is successfully sent or a predetermined number of trials is reached. When $l = 1$, the multiple logical paths are tried in series. Since each time the descendants involved in forwarding are randomly chosen, it is very likely that different logical paths are taken at different times. This method can achieve similar reliability as trying multiple logical paths simultaneously with a larger l. However, it requires other mechanisms such as acknowledgment messages or time-out to detect failed attempts.

1.2.2. *Generalized CBKE scheme*

In the basic CBKE scheme, we consider k as a binary number and change a 1 bit to 0 after each logical hop when forwarding the session key from s to d. We can generalize the scheme by representing k as a radix 2^b number, where $b \geq 1$, and changing a non-zero digit of k to 0 when forwarding the session key to d. In the generalized CBKE scheme, $H(k)$ represents the number of non-zero digits in k.

Key pre-distribution. When the radix is 2^b, k can be represented with g digits $(k_{g-1}, k_{g-2}, \ldots, k_1, k_0)$, where $g = \lceil m/b \rceil$. Each digit k_i, $i = 0, \ldots, g-1$, has 2^b different values. Two nodes preshare a key if their distance is $a \times 2^{ib}$, where $0 < a < 2^b$ and $0 \leq i < g$. In other words, if the distance between two nodes has only one non-zero digit, they preshare a key.

Because k has g digits and each digit has $2^b - 1$ non-zero values, a node needs to establish a unique key with $2 \times (2^b - 1) \times (g - 1) + (2^b - 1)$ nodes. Using the method proposed in [22], each node needs to store $(2^b - 1) \times g + 1 = (2^b - 1) \times \lceil m/b \rceil + 1$ keys.

We use an example to illustrate key pre-distribution in the generalized CBKE scheme. Suppose $N = 256$ and $b = 2$. The distance between two nodes can be represented with a 4-digit radix-4 number. Node 0 preshares a unique key with 21 nodes, which are listed in Table 1.1. In the table, we use a subscript 4 to indicate radix-4 numbers. We also group the nodes according to the position of the non-zero digit in their distance to node 0. In general, the nodes presharing a key with a particular node can be divided into g groups as k has g digits.

Key establishment. The key establishment process in the generalized CBKE is

Table 1.1. Nodes presharing a key with node 0 ($N = 256, b = 2$).

Distance to node 0	Node IDs	Group
$0001_4 = 1$	1 and 255	
$0002_4 = 2$	2 and 254	1
$0003_4 = 3$	3 and 253	
$0010_4 = 4$	4 and 252	
$0020_4 = 8$	8 and 248	2
$0030_4 = 12$	12 and 244	
$0100_4 = 16$	16 and 240	
$0200_4 = 32$	32 and 224	3
$0300_4 = 48$	48 and 208	
$1000_4 = 64$	64 and 192	4
$2000_4 = 128$	128	

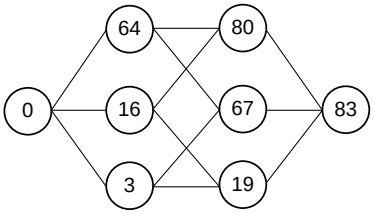

Fig. 1.4. Logical paths from source 0 to destination 83 ($b = 2$).

similar to the process in the basic CBKE, which is shown in Fig. 1.2. The only difference is that at each hop, a node n_i randomly selects a non-zero digit k_u in $k = D(n_i, d)$ and compute $n_{i+1} = n_i \pm k_u \times 2^{bu}$. Similarly, $+$ is performed if n_i is closer to d clockwise. Otherwise, $-$ is performed. After each logical hop, one of the non-zero digits in k is changed to 0. Consequently, a logical path has at most g hops. On average, the larger b is, the shorter logical paths we have.

As in the basic CBKE scheme, we can use l to control the number of descendants to which a node may forward a session key. Also, the multiple paths can be utilized in parallel, in series, or in a combination of both.

We use the example in Fig. 1.4 to illustrate the generalized CBKE scheme. Suppose $N = 256$, $b = 2$, $l = 1$, and node 0 wants to send a session key r to node 83. The distance between the two nodes is $k = D(0, 83) = 83 = 1103_4$. Since k has three non-zero digits, $H(k) = 3$. Node 0 has three descendants: nodes 3, 16, and 64. It randomly picks one of the descendants, say node 3, as the next node on the logical path. After node 3 receives r, it selects either node 67 or node 19 as the next node, which then forwards r to node 83. Because of the larger radix, a logical path now has only three hops, instead of four hops in the case shown in Fig. 1.3.

Relationship to existing deterministic schemes. We remark that several existing deterministic schemes are special cases of CBKE. The full pairwise scheme is CBKE with $b = m$, where each pair of nodes preshare a unique key. PIKE [15] is

essentially a CBKE scheme with $b = m/2$, where each node prestores $O(\sqrt{N})$ keys and each logical path contains two logical hops. When nodes in a group share the same key, CBKE with $b = m$ has a single group and all nodes share the same key, which becomes the single-key predistribution scheme.

1.3. Properties of CBKE Schemes

We next describe several properties of the CBKE schemes. Recall that $k = D(s, d)$, where s is the source node and d is the destination node. We consider k as a radix 2^b number and $H(k)$ is the number of non-zero digits in k.

Memory overhead. As mentioned earlier, the number of preloaded keys into each node is $(2^b - 1) \times \lceil m/b \rceil + 1$, where m is the number of bits required to represent a node ID. Hence the memory overhead increases with b.

Length of a logical path. When the session key is forwarded from s to d, the number of non-zero digits in k is reduced by one after each logical hop. Eventually the distance becomes 0 at d. Therefore, a logical path from s to d has $H(k)$ logical hops. The expectation of $H(k)$ is roughly

$$\lfloor \frac{m-1}{b} \rfloor \times \frac{2^b - 1}{2^b} + \frac{2^a - 1}{2^a}, \tag{1.1}$$

where $a = (m - 1) \mod b$. This is can be explained as follows. For a randomly chosen source-destination pair, their distance is uniformly distributed in $[1, 2^{m-1}]$. Consider all the numbers from 0 to $2^{m-1} - 1$. They can be represented by $(m - 1)$ bits. We divide $(m - 1)$ bits in groups of b bits (since the radix is 2^b). Then for each b bits in a group, the expected Hamming weight is $(1 - 1/2^b)$ since the Hamming weight is not 1 only when all these b bits are zero, which happens with the probability of $1/2^b$. Since there are $\lfloor \frac{m-1}{b} \rfloor$ groups, we obtain the first term in (1.1). The second term in (1.1) represents the expected Hamming weight of the a bits that are not included in the groups. The above does not consider the distance of 2^{m-1}. When considering it, we obtain the exact formula for the expectation of $H(k)$,

$$E(H(k)) = \frac{(\lfloor \frac{m-1}{b} \rfloor \times \frac{2^b-1}{2^b} + \frac{2^a-1}{2^a}) \times 2^m + 1}{2^m - 1}. \tag{1.2}$$

We can see that (1.1) provides a close approximation to (1.2) for relatively large m. Observe from the above that, on average, $H(k)$ and hence the length of logical paths decrease with b. When $b = 1$, the average number of hops is about $(m-1)/2$. It reduces to approximately $3(m - 1)/8$ when $b = 2$, and to approximately 2 when $b = m/2$.

Number of logical paths and hops. There are more than one logical path from s to d if $H(k) > 1$. We now compute the total number of logical paths that can be

used for key establishment. Let S denote the set of nodes that may participate in forwarding and S_i denote the set of nodes in S whose distance to s has i non-zero digits, $i = 0, \ldots, H(k)$. In particular, $S_0 = \{s\}, S_{H(k)} = \{d\}$. Starting from S_0, the session key is forwarded to a node in S_1, and then to a node in S_2, and so on. Consider a node $n_i \in S_i$. We have $D(n_i, d) = H(k) - i$. Therefore, node n_i has $H(k) - i$ descendants, all of which are in S_{i+1}. Node n_i has $(H(k) - i)$ choices when selecting a descendant in S_{i+1}. For example, s has $H(k)$ choices, and a node in S_1 has $H(k) - 1$ choices. Hence, the total number of logical paths from s to d is

$$\prod_{i=0}^{H(k)-1} (H(k) - i) = H(k)!. \tag{1.3}$$

The number of nodes in set S_i is $\binom{H(k)}{i}$ because i out of $H(k)$ non-zero digits in k have changed to 0 for a node in S_i. Combining with the previous observation that each node in S_i has $H(k) - i$ different ways to reach nodes in S_{i+1}, we obtain the total number of logical hops that may be used in forwarding as

$$\sum_{i=0}^{H(k)-1} (H(k) - i)\binom{H(k)}{i} = H(k)2^{H(k)-1}. \tag{1.4}$$

We now look at some examples. In Fig. 1.3, $b = 1$, $S_0 = \{0\}$, $S_1 = \{1, 2, 16, 64\}$, $S_2 = \{3, 17, 18, 65, 66, 80\}$, $S_3 = \{19, 67, 81, 82\}$, and $S_4 = \{83\}$. Each logical path from node 0 to node 83 contains $H(k) = H(01010011_2) = 4$ logical hops. The total number of logical paths from node 0 to node 83 is $H(k)! = 4! = 24$. The total number of logical hops is $H(k)2^{H(k)-1} = 4 \times 2^3 = 32$. In Fig. 1.4, $b = 2$, $S_0 = \{0\}$, $S_1 = \{3, 16, 64\}$, $S_2 = \{19, 67, 80\}$, and $S_3 = \{83\}$. Each logical path from node 0 to node 83 has $H(k) = H(1103_4) = 3$ logical hops. The total number of logical paths from node 0 to node 83 is $H(k)! = 3! = 6$. The total number of logical hops is $H(k)2^{H(k)-1} = 3 \times 2^2 = 12$.

Summary. In summary, when b increases, the memory overhead increases, while, on average, the length of a logical path and the total number of logical paths decrease. If we assume that nodes in a network are uniformly distributed and node failures are also uniformly random, then each logical hop contains a similar number of physical hops and has a similar probability to break. Therefore, on average, a shorter logical path incurs less communication cost and provides better reliability. However, when a logical path is short, the total number of logical paths from a source to a destination is also small, which limits the number of logical paths that can be used for key establishment, and hence limits the degree of reliability that can be achieved. In Section 1.4, we will study how the choice of b affects communication cost and reliability in detail.

1.4. Performance Trade-offs

In this section, we explore how the choices of b and l affect the *memory overhead, reliability*, and *communication cost* of CBKE. The memory overhead is the number of keys that are preloaded into a sensor node. The reliability is the probability that a pair of nodes can establish a session key successfully when nodes may fail. The communication cost is the average number of messages transmitted in the network for a key establishment. In the following, simulation results are obtained using a simulator that we developed.

We assume a network of $N = 4096$ nodes ($m = 12$) and each node has a fixed physical position. The transmission range of a node is one unit and the node density is 20 nodes per square unit. We adopt GPSR [23] as the routing protocol in the physical network. So every node knows the position of the destination node and its neighboring nodes. We set b and l to a wide range of settings: $b = 1, 2, \ldots, 6$, and $l = 1, 2, 3$ and unlimited (i.e., in the flooding mode). Note that when $b = 6$, CBKE is similar to PIKE [15].

We first compare the memory overhead for different choices of b. As we discussed in Section 1.2.2, the number of preshared keys on each node is $(2^b - 1) \times \lceil m/b \rceil + 1$, which increases with b. In particular, when $b = 1$, the number of preshared keys is $O(\log_2 N)$; when $b = m/2$, the number of preshared keys is $O(\sqrt{N})$. Fig. 1.5 plots the memory overhead for various values of b. We observe that, as b increases from 1 to 6, the number of preloaded keys first increases gradually and then increases sharply. Therefore, when memory is a stringent constraint, it is desirable to choose smaller values of b.

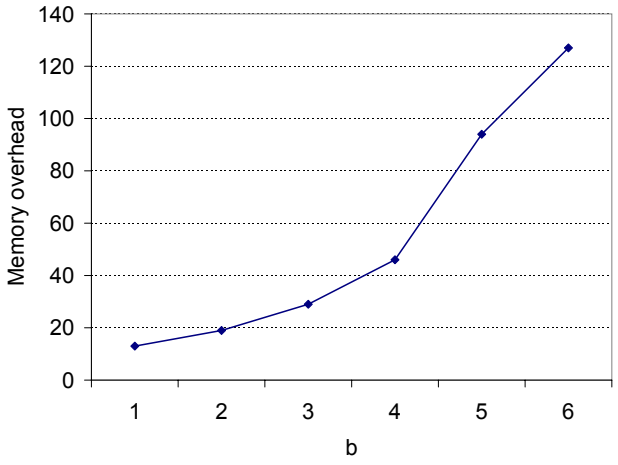

Fig. 1.5. Memory overhead, $m = 12$.

We now consider the average number of logical hops. Fig. 1.6 plots the average

number of hops for all combinations of source and destination pairs versus b. As expected, the average number of hops decreases as b increases. Since the nodes are uniformly randomly deployed, the lengths of logical paths reflects the lengths of physical paths, and hence communication costs. To confirm this, we randomly choose $10,000$ pairs of nodes in the network and establish keys for each pair. We then obtain the average communication cost when b increases from 1 to 6. The communication cost matches the results in Fig. 1.6 (scaled up by the average length of a logical hop).

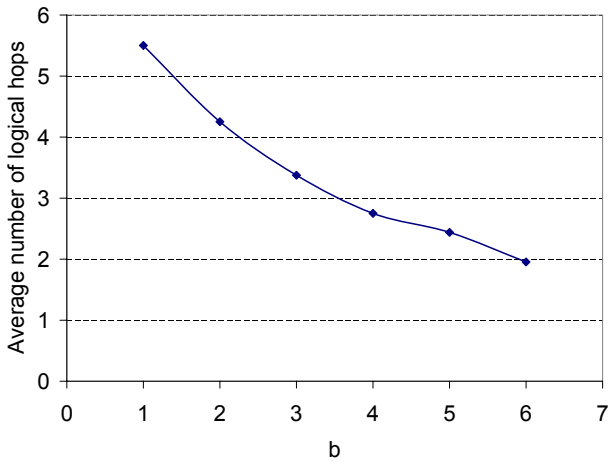

Fig. 1.6. Average number of logical hops, $m = 12$.

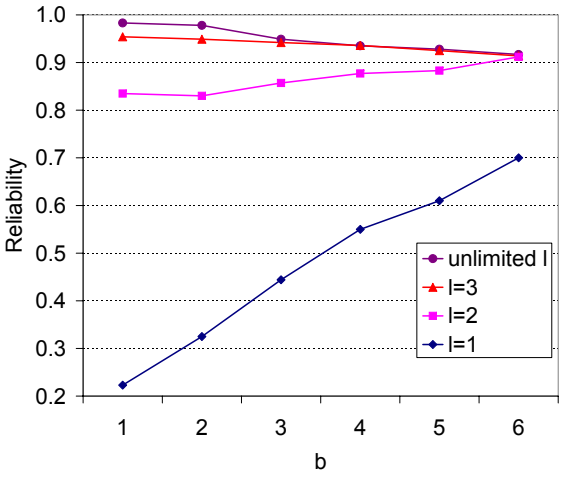

Fig. 1.7. Reliability of CBKE, $p = 0.3$, $m = 12$.

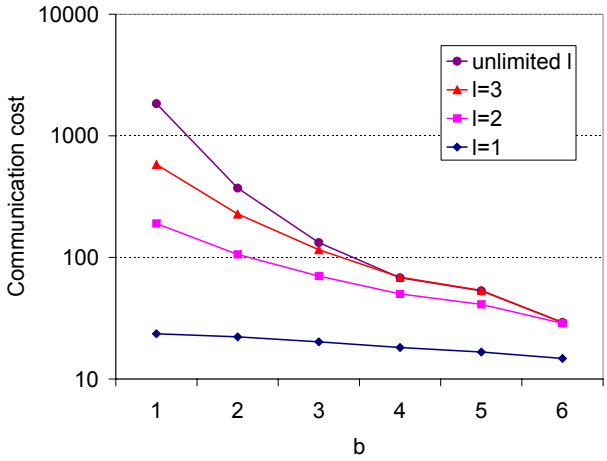

Fig. 1.8. Communication cost of CBKE, $p = 0.3$, $m = 12$.

We now explore how the values of b and l affect the reliability and communication cost of CBKE. We set node failure rate, p, to values between 0.1 and 0.5, with an increment of 0.1. For each value of p, we randomly choose $10,000$ pairs of alive nodes in the network and establish keys for each pair. We then obtain the average reliability and communication cost over all the pairs.

Figures 1.7 and 1.8 plot the average reliability and communication cost for $p = 0.3$. We observe that when $l = 1$, the average reliability increases significantly with b. This is because larger values of b lead to shorter logical paths and thus to higher probabilities of successful key establishments. As l increases from 1 to 2, the average reliability for all values of b increases significantly, indicating the effectiveness of using multiple paths to increase reliability. When $l = 3$, we observe that the average reliability decreases with b. This is because when $l = 3$, more logical paths can be utilized for key establishment for small values of b and hence result in good reliability. As b increases, the number of logical paths that can be utilized for key establishment decreases significantly. Although each logical path is more likely to be successful, the overall reliability decreases because of the small number of logical paths. When not limiting l, we observe a similar trend as when $l = 3$. In Fig. 1.8, as expected, we observe that the average communication cost decreases with b since the average length of a logical path decreases as b increases. Furthermore, as l increases, the average communication cost may increase significantly, especially when b is small, since a much larger number of logical hops are involved.

We observe similar trends in reliability and communication cost for other values of p $(0.1, 0.2, 0.4,$ and $0.5)$. In general, the average reliability increases with b for small values of l, and decreases with b for large values of l; the highest reliability is achieved when $b = 1$ and not limiting l, at the price of the highest communication

cost.

In summary, we see a clear trade-off in memory overhead, reliability, and communication cost when choosing different values of b and l in CBKE. The best choices of the parameters depends on the constraints of a specific setting. For instance, one may determine b and l to achieve the highest reliability for certain constraints of memory overhead and communication cost. By tuning the parameters of b and l, CBKE provides a wide range of performance trade-offs among memory overhead, reliability, and communication cost.

1.5. Security of CBKE

We now analyze the security of CBKE. Under a *passive* attack model where attackers only passively monitor the communications in the sensor networks, CBKE is secure. This is because when a session key is forwarded along a logical path, it is protected with the shared key between two nodes on every hop, and hence cannot be revealed through passive monitoring. Under an *active* attack model where attackers are capable of compromising sensor nodes and then retrieve the information stored on those nodes (including preshared keys), CBKE can become insecure. This is because a session key is known by all intermediate nodes along a logical path (an intermediate node needs to decrypt the protected session key with the shared key on the previous hop and then encrypt it with the shared key on the next hop). Therefore, if an attacker compromises an intermediate node, it can potentially obtain the session key, and decrypt all the messages protected by this session key. We next analyze the security of CBKE assuming an active attack model where a session key and thus the communication channel between a source and destination are compromised if any node on the logical path is compromised.

Assume the logical path from the source node s to the destination d has $H(k)$ hops, where $H(k)$ is the number of non-zero digits in k, and k is the distance between s and d. We also assume that both s and d are secure, and the probability of other nodes being compromised is p_s. Then the probability of the session key being compromised is $1-(1-p_s)^{H(k)-1}$ (since there are $H(k)-1$ intermediate nodes). It is clear that as $H(k)$ decreases, the probability that a session key is compromised decreases. Adopting a larger radix decreases $H(k)$ and thus improves security. This can be seen in Fig. 1.9, which illustrates how the radix affects the security of CBKE with $m = 12$ and $l = 1$. This figure shows the probabilities of the channel between s and d being compromised for four choices of p_s: 0.01, 0.02, 0.05 and 0.1. A point in the figure represents an average probability when k is uniformly randomly chosen from 1 to 2^{m-1} (i.e., s and d are chosen randomly). We can see clearly from the figure that increasing b results in less vulnerability or better security. Intuitively, the less nodes participate in the key establishment process, the less likely the key and the channel are compromised.

For a fixed radix, the security of CBKE can be improved as follows. Source

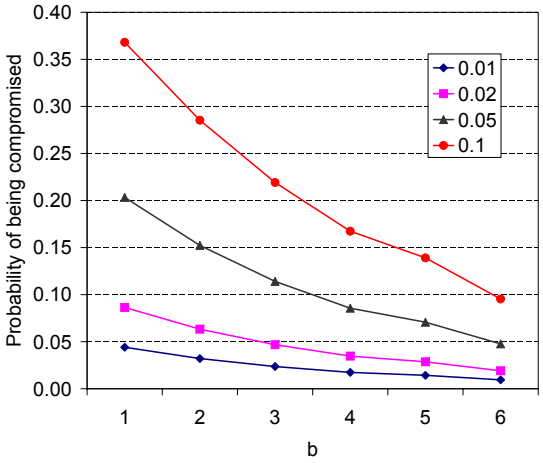

Fig. 1.9. Security of CBKE, $m = 12$, $l = 1$.

node s sends multiple secret values to d through multiple disjoint paths, one value on one path, and an attacker needs to know all the values to construct the session key. To minimize the communication and energy overhead, we look at the case that s sends two secret values to d over two disjoint path. In particular, one value is sent through the clockwise path and the other through the counterclockwise path. Both s and d can construct the session key by exclusive or-ing the two values. Since the two paths are disjoint, none of the intermediate nodes knows both secret values. Even if an adversary compromises one or more nodes on one path, he cannot construct the session key. Assume that the clockwise distance from s to d is k. The counterclockwise distance from s to d is $N - k$, where $N = 2^m$ is the total number of nodes in the networks. Assume that both s and d are secure, and the probability of other nodes being compromised is p_s. Then the probability of the session key being compromised is $(1-(1-p_s)^{H(k)-1})(1-(1-p_s)^{H(N-k)-1})$. We now reexamine the example illustrated in Fig. 1.9, in which $m = 12$ and $l = 1$. Fig. 1.10 shows the security of CBKE when using two disjoint paths. We again consider four choices of p_s: 0.01, 0.02, 0.05, and 0.1. A point in the figure represents an average probability that the channel between s and d is compromised when k is uniformly randomly chosen from 1 to 2^{m-1} (i.e., s and d are chosen randomly). We see from these two figures that the security of CBKE is improved significantly by forwarding secrets through two disjoint paths. For example, when $p_s = 0.1$ and $b = 1$, the probability of compromising the session key reduces from 0.368 (single-path) to 0.152 (two-path). When $p_s = 0.1$ and $b = 6$, the probability of compromising the key decreases from 0.095 (single-path)to 0.010 (two-path).

This improvement, however, comes at a cost. First, sending multiple secrets incurs large communication and energy overhead as more packets are forwarded from a source to a destination. In the case the source sends out two secret values,

the communication cost is doubled on average. Secondly, sending multiple secrets reduces reliability. When one of the multiple paths fails to deliver the secret to the destination, the destination cannnot construct the session key and thus cannot establish a secure communication channel with the source.

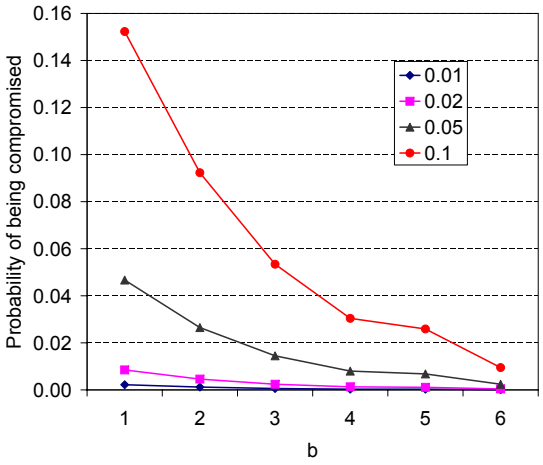

Fig. 1.10. Security of CBKE when sending two secrets through two disjoint paths, $m = 12$.

1.6. Further Improvements to CBKE

CBKE can be further improved, especially under specific configurations. We discuss a few of them here.

1.6.1. *Forwarding paths*

In CBKE, the total number of logical paths from s to d is $H(k)!$. For relatively small $H(k)$ (e.g., $H(k) = 1$ or 2), $H(k)!$ logical paths may not provide sufficient amount of path redundancy. More logical paths can be utilized if one chooses additional intermediate nodes. For instance, suppose $s = 0$, $d = 6$, $m = 4$, and $b = 1$. The current CBKE scheme provides only two paths: $(0, 2, 6)$ or $(0, 4, 6)$. If both nodes 2 and 4 fail, the session key cannot be forwarded to node 6. However, it can be done through additional paths such as $(0, 8, 6)$ and $(0, 1, 5, 6)$.

One way to choose additional forwarding paths is by representing the distance between a source and a destination using non-adjacent form (NAF). NAF is a signed-digit representation that does not allow adjacent non-zero digits. For instance, in the example above, the distance 6 can be represented in its NAF of $(1, 0, -1, 0)_{NAF}$, which provides two additional forwarding paths: $(0, 8, 6)$ and $(0, 14, 6)$. One advantage of using NAF is that the distance thus represented may have a smaller

Hamming weight, leading to a shorter logical path from a source to a destination. For instance, consider source 0 and destination 7. In regular binary form, the distance from the source to the destination, 7, has Hamming weight of 3, and hence the number of logical hops is 3. In NAF, the Hamming weight reduces to 2 (the NAF of 7 is $(1, 0, 0, -1)_{NAF}$), and hence the number of logical hops reduces to 2. When m is large and $b = 1$, the average number of hops between two nodes reduces from $(m-1)/2$ (in regular binary form) to about $(m-1)/3$ when using NAF (since the Hamming weight of an integer using NAF is one third of the number of bits in the integer [24]). The binary NAF provides even shorter logical paths compared to the case of $b = 2$ (in regular form), in which the average number of hops is about $3(m-1)/8$ (see Section 1.3).

1.6.2. *Partial deployment*

So far, we have assumed that the number of nodes in the network is a power of the radix selected. In a real deployment, the assumption may not be true. Nodes deployed in a sensor network may not take all the IDs in the ID space. We will refer to this scenario as *partial deployment*. Since some IDs are not present in the network, they cannot participate in the key forwarding as expected. A simple approach is to consider these IDs belong to failed nodes and use a trial-and-error strategy. This approach, however, has large communication overhead. It will be more efficient to let the nodes be aware of undeployed IDs and take this into account when deciding forwarding paths. We next describe a scheme for partial deployment. For simplicity, we take the case of $b = 1$ as an example; our scheme also applies to other radices. Note that when $b = m/2$, CBKE is similar to PIKE. For this specific case, the partial deployment problem is also discussed in [15].

Let us assume we represent an ID with m bits. In the case of partial deployment, we deploy IDs in the order that is specified by a counter. This counter starts with 0. It is similar to a binary counter, but it adds one to the most significant bit (MSB) and propagates the carry from MSB to the least significant bit (LSB). We next use $m = 4$ as an example to illustrate this deployment order. The first node deployed has ID 0. Adding one to the MSB, the second node to be deployed is hence $1000_2 = 8$. Again adding one to the MSB and propagating the carry from MSB to LSB, the third node to deployed is hence $0100_2 = 4$. Afterwards, node $1100_2 = 12$ is deployed, followed by $0010_2 = 2$, $1010_2 = 10$, $0110_2 = 6$, $1110_2 = 14$, $0001_2 = 1$, $1001_2 = 9$, $0101_2 = 5$, $1101_2 = 13$, $0011_2 = 3$, $1011_2 = 11$, $0111_2 = 7$, and $1111_2 = 15$. We can see, in this deployment order, a newly deployed ID preshares a unique key with at least two IDs that have already been deployed (if more than two IDs have been deployed).

Suppose we have deployed N nodes and $N < 2^m$. Given any two nodes, we look at how to find a logical path between them such that the logical path consists of deployed nodes only. If $N = 2^x$ and $x < m$, the lower $(m-x)$ bits of all the deployed IDs are 0. We can therefore consider the deployed nodes as having ID lengths of

x (i.e., ignore the lower $(m-x)$ bits since they are 0) and take the normal CBKE forwarding strategy. The number of hops on a logical path is no more than $x-1$. If $2^x < N < 2^{x+1}$, the lower $(m-x-1)$ bits of all the deployed IDs are 0. We classify the IDs into two groups, P_x and Q_x, depending on whether bit $(m-x-1)$ is 0 or 1[d]. All the IDs in P_x have bit $(m-x-1)$ as 0; all the IDs in Q_x have bit $(m-x-1)$ as 1. There are 2^x IDs in P_x, represented as $v \times 2^{m-x}$ where $v = 0, 1, 2, \ldots, 2^x - 1$ (since their lower $(m-x)$ bits are 0). There are $(N-2^x)$ nodes in Q_x, having the form of $v \times 2^{m-x} + 2^{m-x-1}$ and $v \in [0, 2^x - 1]$. Suppose that the source node is s and the destination is d. We consider the following four cases:

- Both s and d are in P_x. Node s can take the normal forwarding strategy. The longest path has $x-1$ logical hops.
- Node s is in P_x and d is in Q_x. Suppose d's ID is $v_d \times 2^{m-x} + 2^{m-x-1}$. Node s can first send the key to one node in P_x that preshare a key with d (e.g., node $v_d \times 2^{m-x}$). That node then forwards the session key to d. The longest path in this case consists of x logical hops.
- Node s is in Q_x and d is in P_x. This case is similar to the one above. Node s first sends the session key to a node in P_x that preshares a key with it. That node then forwards the key to d. The longest path in this case has x logical hops.
- Both s and d are in Q_x. We next show that there exists a logical path between s and d that involves only the nodes in Q_x. We consider only the $N-2^x$ nodes in Q_x (i.e., we ignore the nodes in P_x). Let $N_1 = N - 2^x$. Suppose $2^{x_1} \leq N_1 < 2^{x_1+1}$. Since all the IDs in Q_x have the same lower $(m-x)$ bits, we only need to consider the upper x bits. Our problem of finding a logical path reduces to one that considers a smaller number of nodes (i.e., the number of nodes reduces from N to N_1) and a smaller ID space (from x bits to x_1 bits). In this new problem, we repeat the process as in the original problem until the relationship of s and d falls into one of the three earlier cases, and we are done. The logical path therefore only contains nodes in Q_x, and the longest path consists of no more than x_1 logical hops ($x_1 < x$).

To summarize, for any value of N, we can always find a logical path between any two nodes and the logical path consists of only deployed nodes. Although we take $b = 1$ as an example to illustrate the method that handles partial deployment, it can be adopted for handling other radices when $b > 1$.

[d]Recall that we refer to the lowest bit (the right most bit) as bit 0, the second lowest bit as bit 1, the third lowest bit as bit 2, and so on.

1.7. Conclusions

In this chapter, we have discussed CBKE, a pairwise key establishment protocol for large-scale sensor networks. The scheme provides more flexibility than previously proposed deterministic key exchange protocols to balance memory overhead, reliability, and communication cost. We also evaluated the trade-offs using simulation. Last, we analyzed the security of CBKE, and proposed further improvement to CBKE.

References

[1] F. Zhang, Z. Shi, and B. Wang. Chord-based key establishment schemes for sensor networks. In *Proceedings of ITNG*, pp. 731–737, Las Vegas, NV (April, 2008).
[2] S. A. Çamtepe and B. Yener. Key distribution mechanisms for wireless sensor networks: a survey. Technical Report TR-05-07, Computer Science Department, Rensselaer Polytechnic Institute (March, 2005).
[3] W. Zhang, M. Tran, S. Zhu, and G. Cao. A random perturbation-based scheme for pairwise key establishment in sensor networks. In *ACM MobiHoc*, Montreal, QC, Candada (September, 2007).
[4] L. Zhou, J. Ni, and C. V. Ravishankar. Efficient key establishment for group-based wireless sensor networks. In *Proc. of ACM Workshop on Wireless Security (WiSe)* (September, 2005).
[5] D. Liu, P. Ning, and W. Du. Group based key pre-distribution in wireless sensor networks. In *Proc. of ACM Workshop on Wireless Security (WiSe)* (September, 2005).
[6] S. Zhu, S. Setia, and S. Jajodia, Leap: Efficient security mechanisms for large-scale distributed sensor networks, *ACM Transactions on Sensor Networks (TOSN)*. **2**(4), 500–528 (November, 2006).
[7] D. Liu and P. Ning. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In *Proc. of the 10th Annual Network and Distributed System Security Symposium*, pp. 263–276 (February, 2003).
[8] D. Liu and P. Ning, Multilevel μ-TESLA: Broadcast authentication for distributed sensor networks, *ACM Transactions on Embedded Computing Systems (TECS)*. **3**(4), 800–836 (November, 2004).
[9] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *MobiCom*, pp. 189–199 (July, 2001).
[10] L. Eschenauer and V. Gligor. A key management scheme for distributed sensor networks. In *ACM Conference on Computer and Communication Security (CCS)*, pp. 41–47 (November, 2002).
[11] A. P. H. Chan and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, pp. 197–213 (May, 2003).
[12] W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *ACM Conference on Computer and Communication Security (CCS)*, pp. 42–51, New York, NY (October, 2003).
[13] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *ACM Conference on Computer and Communication Security (CCS)*, pp. 52–61, New York, NY (October, 2003).
[14] P. Tague and R. Poovendran, A canonical seed assignment model for key predis-

tribution in wireless sensor networks, *ACM Transactions on Sensor Networks*. **3**(4) (October, 2007).

[15] H. Chan and A. Perrig. PIKE: Peer intermediaries for key establishment in sensor networks. In *IEEE INFOCOM* (March, 2005).

[16] W. Du, J. Deng, Y. S. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *Proc. IEEE INFOCOM*, Hong Kong (March, 2004).

[17] F. Liu and X. Cheng. A self-configured key establishment scheme for large-scale sensor networks. In *Proc. of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)* (October, 2006).

[18] L. Ma, X. Cheng, F. Liu, J. Rivera, and F. An, iPAK: An in-situ pairwise key bootstrapping scheme for wireless sensor networks, *IEEE Transactions on Parallel and Distributed Systems (TPDS)*. **18**(8), 1174–1184 (August, 2007).

[19] F. Liu and X. Cheng, LKE: A self-configuring scheme for location-aware key establishment in wireless sensor networks, *IEEE Transactions on Wireless Communications (TWC)*. **7**(1), 224–232 (January, 2008).

[20] A. Y. Teymorian, L. Ma, and X. Cheng. CAB: A cellular automata-based key management scheme for wireless sensor networks. In *Military Communications Conference (MILCOM)* (October, 2007).

[21] I. Stoica, R. Morris, D. L. Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. In *SIGCOMM*, pp. 149–160, San Diego, CA (August, 2001).

[22] S. I. Huang. Adaptive random key distribution schemes for wireless sensor networks. In *Proceedings of the International Workshop on Advanced Developments in Software and Systems Security* (December, 2003).

[23] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MobiCom*, pp. 243–254 (August, 2000).

[24] G. Reitwiesner, Binary arithmetic, *Advances in Computers*. **1**, 231–308, (1960).