# Memory-Efficient Hypercube Key Establishment Scheme for Micro-Sensor Networks

Kyung-suk Lhee

*ABSTRACT—A micro-sensor network is comprised of a large number of small sensors with limited memory capacity. Current key-establishment schemes for symmetric encryption require too much memory for micro-sensor networks on a large scale. In this paper, we propose a memory-efficient hypercube key establishment scheme that only requires logarithmic memory overhead.*

*Keywords— Sensor network, symmetric key cryptography, key pre-distribution, hypercube.*

## I. Introduction

A micro-sensor network is comprised of a large number of small sensors, which have limited computation capability and memory capacity. The current representative sensor is Mica2, which has 128 kB memory. There are, however, much smaller low-power micro-sensors with far less memory (Table 1). Those sensors are designed for small footprint and low-power components, which necessitate small memory.

Table 1. Examples of micro-sensors.

| | Memory (kB) | Size (mm$^2$) | Energy/Inst. (pJ) |
|---|---|---|---|
| Hempstead et al. [1] | 2 | unknown | < 100 |
| Spec [2] | 3 | 2.5 × 2.5 | < 83 |
| SNAP [3] | 8 | unknown | < 75 |
| Smart Dust [4] | 17 | 16 × 16 | 14 |
| Mica2 | 128 | 58 × 32 | 1,500 |

Manuscript received Oct. 10, 2007; revised Feb. 29, 2008.

This work was supported by the new faculty research fund of Ajou University, Rep. of Korea, in 2005.

Kyung-suk Lhee is with the School of Information and Computer Engineering, Ajou University, Suwon, Rep. of Korea.

In this paper, we propose a memory-efficient hypercube symmetric key establishment scheme that only requires a logarithmic number of pre-distributed keys per node; therefore, it is suitable for micro-sensors deployed on a large scale. Our scheme does not have the security issues found in SPINS [5], and it has much lower memory overhead than PIKE [6], which currently has the lowest memory overhead next to SPINS.

## II. Related Work

SPINS [5] pre-distributes a unique, pair-wise key to each node for secure communication with the base station. After deployment, each node uses the base station as the trusted intermediary to establish pair-wise keys with other nodes within its signal range (called *path-keys*). This scheme only requires a single pre-distributed key per node, but it is not a robust scheme because the base station becomes the single point of attack. During the key establishment process, the communication load is focused around the base station, so an adversary may be able to locate the base station via traffic analysis. Moreover, the nodes close to the base station suffer from higher energy consumption [6]. For this reason, a more secure approach is needed.

PIKE [6] regards the node space as a virtual two-dimensional mesh and assigns each node an ID ($x$, $y$) according to its $x$ and $y$ coordinates. PIKE pre-distributes keys such that each node shares pair-wise keys with all the nodes along the same row and the same column ($2(\sqrt{n}-1)$ pre-distributed keys per node). After deployment, each node exchanges path-keys with other nodes in the signal range using one node as the secure intermediary. For example, in Fig. 1(a), node (1, 1) can establish a path-key with (2, 2) via (1, 2) as follows.

**Step 1.** (1, 1) encrypts and sends a path-key to (1, 2), a secure intermediary. K(1, 1)(1, 2) denotes the pre-distributed, pair-

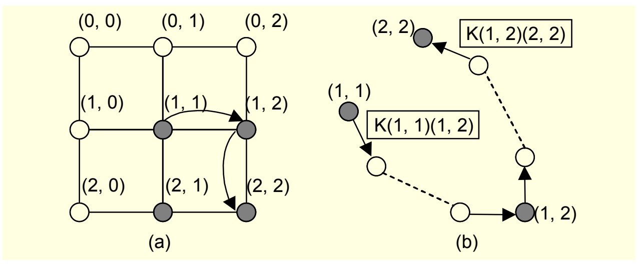

Fig. 1. (a) Sending a path-key in PIKE and (b) the actual path-key route in the network.

wise key used for encrypting the path-key.

**Step 2.** (1, 2) decrypts the path-key, re-encrypts it using K(1, 2)(2, 2), and sends it to (2, 2).

The mesh topology is *virtual*; sensors are actually deployed randomly. For example, (1, 1) may actually be many hops away from (1, 2) as in Fig. 1(b).

PIKE 3D is a natural extension of the basic 2D scheme described above.

## III. Proposed Hypercube Scheme

Our scheme is comparable to PIKE, but the main difference is that our scheme regards the sensor space as a virtual hypercube instead of a two-dimensional mesh. For simplicity, let us assume that the number of sensor nodes $n$ is a power of two ($n$=$2^d$). Each node is labeled with a $d$-bit binary number according to its dimension. Each node shares $d$ pair-wise pre-distributed keys with its virtually connected nodes in the hypercube. After the network is deployed, each node discovers other nodes within its signal range and establishes path-keys with them, using its virtual neighbors as intermediaries. For example, the node (010) shown in Fig. 2(a) has three pre-distributed pair-wise keys for encrypted communication with (000), (011), and (110).

A path-key is delivered by the virtual neighbors using standard hypercube routing. For example, in Fig. 2(a), node (010) can send a path-key to (101) via (011) and (111). Nodes (011) and (111) decrypt and re-encrypt the path-key along the way. Like PIKE, the hypercube topology is virtual. For

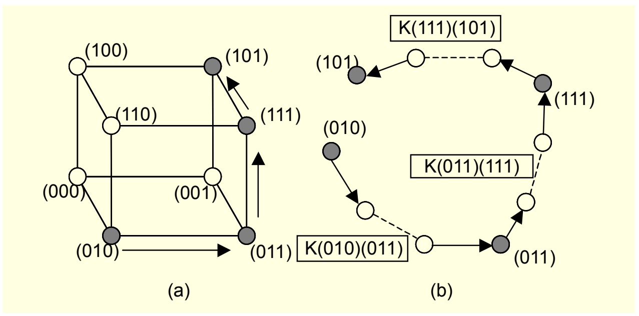

Fig. 2. (a) Sending a path-key in 3D hypercube and (b) the actual path-key route in the network.

instance, (010) may actually be many hops away from (011) as in Fig. 2(b).

### 1. Memory Overhead

Table 2 shows how we calculated the memory overhead of PIKE and our scheme, where $m$ is the key size, and $\log_2 n$ is the number of bits required to encode a node ID. Figure 3 compares the memory overhead where $m$=128. We assumed TinyOS as the operating system, the core components of which require only 400 bytes. The results of the analysis are summarized as follows.

- PIKE 2D/3D cannot be used with Hempstead and Spec.
- PIKE 2D cannot be used with SNAP beyond 900 nodes or with Smart Dust beyond 4,000 nodes.
- PIKE 3D cannot be used with SNAP beyond 8,000 nodes. It can be used with Smart Dust but takes up a significant proportion of the total memory.
- Our scheme takes minimal memory space (< 2 kB); therefore, it can be used for any type of sensors throughout the given range.

### 2. Communication Overhead

Communication overhead of sending one path-key is calculated as the number of hops (approximated as $p$)

Table 2. Memory overhead per node.

| | Node degree | Memory overhead |
|---|---|---|
| PIKE 2D | $2(\sqrt{n}-1)$ | $(m+\log_2 n)\times 2(\sqrt{n}-1)$ |
| PIKE 3D | $3(\sqrt[3]{n}-1)$ | $(m+\log_2 n)\times 3(\sqrt[3]{n}-1)$ |
| Hypercube | $\log_2 n$ | $(m+\log_2 n)\times\log_2 n$ |

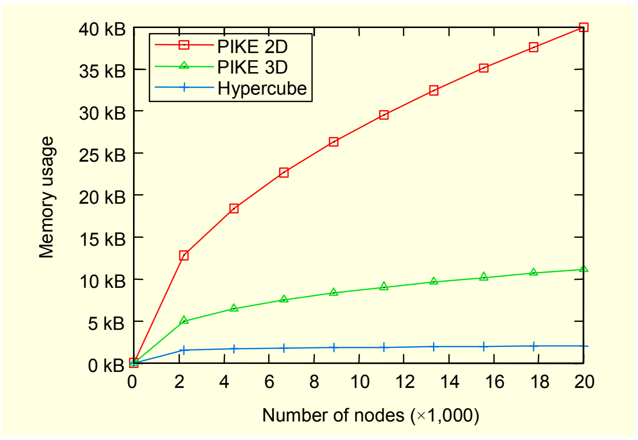

Fig. 3. Memory overhead.

multiplied by the average path length of hypercube $2c\sqrt{n}/3$, which is the expected hop distance between any two nodes [6], where $c$ is some constant. In this paper we assumed that $c$=1.

We calculate $p$ as the sum of the path lengths from one node to all the other nodes, divided by the total number of nodes: $p=\frac{1}{n}\sum_{i=1}^{d} i\times a_i$, where $a_i$ is the number of nodes that are $i$ hops away ($a_i$ is a binomial coefficient). Therefore,

$$p=\frac{1}{n}\sum_{i=1}^{d} i\times\binom{d}{i}=\frac{1}{n}\times d\times 2^{d-1}=\frac{\log_2 n}{2}.$$

Table 3 shows how we calculate communication overhead of PIKE and our scheme, where $v$ is the number of nodes within the signal range of each node (the neighborhood size). We assume that the neighborhood size is constant. Figure 4 compares the communication overhead where the number of neighbors of a node is 60.

Table 4 shows the energy used by each node for communication when the total number of nodes is 20,000, in which case, each node generates 40,000 128-bit path-key transmissions as shown in Fig. 4. For Hempstead and SNAP sensors, which specify their processor architectures only, we assume that the Chipcon CC1101 transceiver is used by many sensors. Table 4 shows that each node consumes only a small fraction of the battery power due to the communication overhead (less than 0.06% of the total battery capacity in our analysis).

Table 3. Communication overhead per node.

| | Ave. path length | Comm. overhead |
|---|---|---|
| PIKE 2D | 2 | $2\times\frac{2\sqrt{n}}{3}\times v$ |
| PIKE 3D | $3\{1-\frac{1}{\sqrt[3]{n}+2}\}$ | $3\{1-\frac{1}{\sqrt[3]{n}+2}\}\times\frac{2\sqrt{n}}{3}\times v$ |
| Hypercube | $\frac{\log_2 n}{2}$ | $\frac{\log_2 n}{2}\times\frac{2\sqrt{n}}{3}\times v$ |

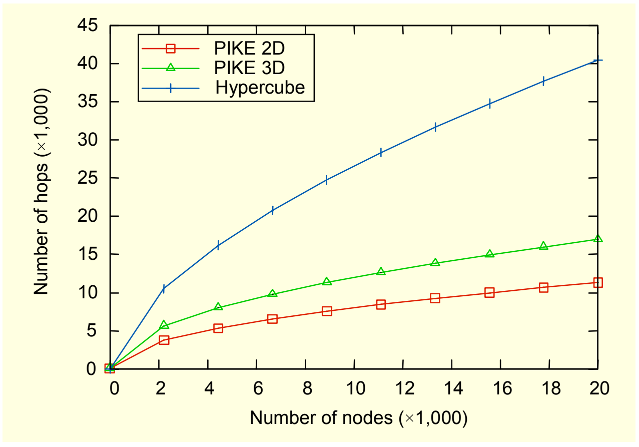

Fig. 4. Communication overhead.

Table 4. Energy used in communication.

| | Chipcon | Spec | Smart Dust |
|---|---|---|---|
| Receive a bit | 50.4 nJ | 9,000 pJ | 12 pJ |
| Send a bit | 54 nJ | 100 pJ | 16 pJ |
| Battery capacity | 890 J (Panasonic SR44 silver oxide coin cell) | 890 J (SR44) | 1 J (radioisotope thermoelectric generator) |
| Energy used per node | 534 mJ | 46.592 mJ | 0.143 mJ |
| Battery used | 0.06% | 0.005% | 0.014% |

## IV. Conclusion

Our scheme can be used for deploying micro-sensors on a large scale due to its minimal logarithmic memory requirement. Unlike SPINS, our scheme does not have a focused communication pattern. Path-key communications are diffused to all nodes; therefore, our scheme is much more resilient to attacks during key establishment.

Our scheme incurs more communication overhead than PIKE does. For micro-sensors, however, the energy consumed by path-key communication is only a small fraction of the battery capacity. Since key establishment is a one-time cost, this communication overhead would not be a major concern. Moreover, a device can harvest energy from nearby light, vibration, heat, and so on [4].

## References

[1] M. Hempstead et al., "An Ultra Low Power System Architecture for Sensor Network Applications," *Proc. ISCA*, 2005, pp. 208-219.

[2] J.L. Hill, *System Architecture for Wireless Sensor Networks*, doctoral dissertation, UC Berkeley, 2003.

[3] V. Ekanayake et al., "An Ultra Low-Power Processor for Sensor Networks," *Proc. ASPLOS*, 2004, pp. 27-36.

[4] B.A. Warneke, *Ultra-Low Energy Architectures and Circuits for Cubic Millimeter Distributed Wireless Sensor Networks*, doctoral dissertation, UC Berkeley, 2003.

[5] A. Perrig et al., "SPINS: Security protocols for sensor networks," *Proc. MobiCom*, 2001, pp.189-199.

[6] H. Chan and A. Perrig, "PIKE: Peer Intermediaries for Key Establishment in Sensor Networks," *Proc. INFOCOM*, 2005, pp. 524-535.