I. J. Communications, Network and System Sciences, **2009, 2, 91-168**
Published Online May 2009 in SciRes (http://www.SciRP.org/journal/ijcns/).

Hierarchical Hypercube Based Pairwise Key Establishment Scheme for Sensor Networks

Lei WANG[1,2]
[1]*College of Software, Hunan University, Changsha, Hunan, China*
[2]*Department of Computer Science, Lakehead University, Thunder Bay, Canada*
Email: *wanglei@hnu.cn*

Received November 27, 2008; *revised January* 17, 2009; *accepted February* 12, 2009

Abstract

Security schemes of pairwise key establishment, which enable sensors to communicate with each other securely, play a fundamental role in research on security issue in wireless sensor networks. A general framework for key predistribution is presented, based on the idea of KDC (Key Distribution Center) and polynomial pool schemes. By utilizing nice properties of *H*2 (Hierarchical Hypercube) model, a new security mechanism for key predistribution based on such model is also proposed. Furthermore, the working performance of tolerance resistance is seriously inspected in this paper. Theoretic analysis and experimental figures show that the algorithm addressed in this paper has better performance and provides higher possibilities for sensor to establish pairwise key, compared with previous related works.

Keywords: Pairwise Key, Sensor Networks, Key Pool, Key Predistribution, H2 Framework

1. Introduction

The security issue in wireless sensor networks has become research focus because of their tremendous application available in military as well as civilian areas. However, constrained conditions existent in such networks, such as hardware resources and energy consumption, have made security research more challenging compared with that in traditional networks.

Current research focus on such security schemes as authentication and key management issues, which are essential to provide basic secure service on sensor communications. Pairwise key establishment enables any two sensors to communicate secretly with each other. However, due to the characteristics of sensor nodes, it is not feasible to utilize traditional pairwise key establishment schemes.

Eeschnaure *et al*. [1] presented a probablitic key predistribution scheme for pairwise key establishment. This scheme picks a random pool (set) of keys *S* out of the total possible key space. For each node, *m* keys are randomly selected from the key pool *S* and stored into the node's memory so that any two sensors have a certain probability of sharing at least one common key. Chan [2] presented two key predistribution techniques: q-composite key predistribution and random pairwise keys scheme. The q-composite scheme extended the performance provided by [1], which requires at least *q* predistributed keys any two sensor should share. The random scheme randomly pickes pair of sensors and assigns each pair a unique random keys. Liu *et al.* [3] developed the idea addressed in previous works and proposed a general framework of polynomial pool-based key predistribution. Based on such a framework, they presented random subset assignment and hypercube-based assignment for key predistribution.

However, it still requires further research on key predistribution because of deficiencies existent in those previous works. Since sensor networks may have dramatic varieties of network scale, the *q-composite* scheme would fail to secure communications as a small number of nodes are compromised. The random scheme may requires each sensor to store a large number of keys, which would be contradicted with hardware constraints of sensor nodes. The random subset assignment would not ensure any two nodes to establish a key path if they do not share a common key. Though the hypercube-based assignment can make sure that there actually exist a key path, however, the possibilities of direct pairwise key establishment are not perfect, leading to large communication overhead.

In order to improve possibilities of direct pairwise key establishment, and depress communication overhead on indirect key establishment, we propose a *H*2 (Hierarchical Hypercube) framework, combined with a new key predistribution scheme. Moreover two new fault tolerance model and corresponding indirect pairwise key establishment schemes are also proposed, by applying nice properties on tolerance resistance *H*2 model has provided. The schemes has better working performance on probabilities of pairwise key establishment between any two sensors.

2. Preliminaries

2.1. Notations and Definitions

Definition1(key predistribution): Cryptographic algorithms are pre-loaded in sensors before node deployment phase.

Definition2 (pairwise key): When any two nods share a common key denoted as *E*, we call that the two nodes share a pairwise key *E*.

Definition 3 (key path): Given two nodes A_0 and A_k, which do not share a pairwise key, if there exists a path in sequence described as $A_0, A_1, A_2, \ldots\ldots, A_{k-1}, A_k$ and any two nodes A_i, A_j ($0 \leqslant i \leqslant k-1, 1 \leqslant j \leqslant k$) share at least one pairwise key, we call that path as a key path.

Definition 4 (*n*-dimensional hypercube interconnection network): *n*-dimensional hypercube interconnection network H_n (abbreviation as *n-cube*) is a kind of network topology that has the following characteristics: 1) It is consisted with 2^n nodes and $n \cdot 2^{n-1}$ links; 2) Each node can be coded with a different binary string with *n* bits such as $b_1 b_2 \ldots b_n$; 3) For any pair of nodes, there is a link between them if there is just one bit different between their corresponding binary strings.

Figure 1 illustrates the topology of a 4-dimensional hypercube interconnection network, which is consisted with 2^4=16 nodes and $4 \cdot 2^{4-1}$=32 links. And the nodes are coded from 0000 to 1111.

2.2. Related Works [1−3]

2.2.1. Polynomial-based Key Predistribution

In the scheme of polynomial-based key predistribution, the key setup server randomly generates a *t*-degree bivariate polynomial $f(x,y) = \sum_{i,j=0}^{t} a_{ij} x^i y^j$ over a finite field F_q, Notes that *q* is fairly large prime number and for any variables *x* and *y*, $f(x,y) = f(y,x)$ is always held. Then the key server computes a share of $f(x,y)$, denoted as $f(i,y)$ for each node, where *i* is assumed to be a unique ID for any sensor node. Every node is pre-loaded with its own share before node-deployment phase. Thus for any two nodes *i* and *j*, node *i* can compute the common key $f(i,j)$ by evaluating $f(i,y)$ at point *j*, and vice visa.

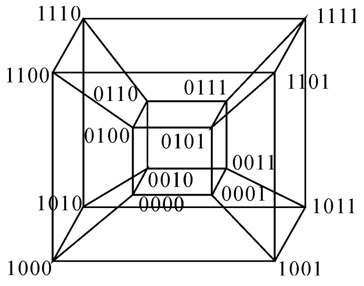

Figure 1. A 4-dimensional hypercube interconnection network.

To predistribute pairwise key with such a scheme as addressed above, node *i*'s storage overhead includes two parts: One is $(t+1)\log q$ storage space for storing a *t*-degree polynomial $f(i,y)$, the other is the storage space for its own ID information. [4] shows that this scheme has ability of *t*-collusion resistant. That is, if there exists no more than *t* compromised nodes in the network, the scheme can ensure the pairwise key is secure between any two normal nodes.

2.2.2. Polynomial Pool-based Key Predistribution

Pairwise key establishment in this scheme is processed in the following three phase: polynomial pool generation and key predistribution, direct key establishment, and path key establishment.

1) polynomial pool generation and key predistribution: This phase is mainly concerned with *t*-degree bivariate polynomial pool (F) generation over a finite field F_q. Then a subset $F_i \in F$ is selected and the shares of all of the polynomials in this subset are assigned to the node *i*.

2) direct key establishment: Assume that node *i* and *j* wants to establish pairwise key, if they have a common share on a same polynomial, they can establish pairwise key by utilizing the polynomial-based scheme. This phase is performed as follows: node *i* may broadcast an encryption list, α, $E_{K_v}(\alpha)$, $v=1,2,\ldots,|F_i|$, where K_v is the share of the *vth* polynomial at point *j*. If node *j* can decrypts any one of these correctly, that means there exists a common share between the two nodes.

3) path key establishment: If there no pairwise key existent between node *i* and *j*, it's necessary to find a key path defined in Definition3. Then the two nodes transmits secret information for pairwise key generation on

this path.

2.2.3. Random Subset Assignment and Hypercube-Based Assignment

1) Random Subset Assignment: Different from polynomial scheme, the main idea of this assignment is to pick a random subset of polynomial pool, denoted as $F_i \in F$, and assign the share of this subset to node *i*.

2) Hypercube-based Assignment: Based on the concept of random subset, this assignment generates polynomial pool by utilizing hypercube model, and assign subsets to nodes according to node's ID.

3. H2 (Hierarchical Hypercube) Model

Definition 5 (*H*2 diagram): Assume that there exist 2^n nodes, the construction algorithm of *n*-dimension *H*2(*n*) is illustrated as follows:

1) Each $2^{\lceil n/2 \rceil}$ nodes are connected as a $\lceil n/2 \rceil$ dimensional hypercube, in which nodes are coded from $\underbrace{00...0}_{\lceil n/2 \rceil} - \underbrace{11...1}_{\lceil n/2 \rceil}$, and such kind of node code is called Inner-Hypercube-Node-Code. As a result, $2^{\lceil n/2 \rceil}$ different such kind of $\lceil n/2 \rceil$ dimensional hypercubes can be formed, where $\lfloor \ \rfloor$ represents the upper integer operation, and $\lceil \ \rceil$ means the lower integer operation.

2) The obtained $2^{\lceil n/2 \rceil}$ different such kind of $\lceil n/2 \rceil$ dimensional hypercubes are codes from $\underbrace{00...0}_{\lceil n/2 \rceil} - \underbrace{11...1}_{\lceil n/2 \rceil}$, and such kind of node code is called Outer-Hypercube-Node-Code. And then, the nodes in the $2^{\lceil n/2 \rceil}$ different such kind of $\lceil n/2 \rceil$ dimensional hypercubes with the same Inner-Hypercube-Node-Code are connected as a $\lceil n/2 \rceil$ dimensional hypercube, so we can obtain $2^{\lceil n/2 \rceil}$ different such kind of $\lceil n/2 \rceil$ dimensional hypercubes.

3) The graph constructed through the above two steps is called a *H*2 graph. And it is obvious that each node in the H2 graph is coded as (*r*,*h*), where *r* ($\underbrace{00...0}_{\lceil n/2 \rceil} \leqslant r \leqslant \underbrace{11...1}_{\lceil n/2 \rceil}$) is the node's Inner-Hypercube-Node-Code, and *h* ($\underbrace{00...0}_{\lceil n/2 \rceil} \leqslant h \leqslant \underbrace{11...1}_{\lceil n/2 \rceil}$) is the node's Outer- Hypercube-Node-Code.

Theorem 1: There exist 2^n in *H*2(*n*) diagram.

Proof: The conclusion is naturally held as $2^n = 2^{\lceil n/2 \rceil} * 2^{\lceil n/2 \rceil}$.

Theorem 2: The diameter of *H*2 (*n*) is *n*.

Proof: As the diameter of $\lceil n/2 \rceil$ dimension hypercube is $\lceil n/2 \rceil$, and it is naturally held for the case of $\lceil n/2 \rceil$ dimension hypercube. Thus the diameter of *H*2(*n*) is $\lceil n/2 \rceil + \lceil n/2 \rceil = n$ according to definition5.

Theorem 3: The distance of any two nodes $A(r_1,h_1)$ and $B(r_2,h_2)$ in *H*2(*n*) is expressed as $d\ (A,B) = d_h(r_1, r_2) + d_h(h_1, h_2) + 1$ where d_h is *Hamming* distance.

Proof: Since the distance of any two nodes is the *Hamming* distance of their corresponding codes, it is held according to definition5.

4. Pairwise Key Establishment Scheme Based on H2 Model

As addressed above, polynomial-based and polynomoial-based schemes have some limitations. In this section we propose a new pairwise key establishment and predistribution scheme based on H2 model. The new algorithm is composed of three phases: polynomial pool generation and key predistribution, direct key establishment, and path key establishment.

4.1. Polynomial Pool Generation and Key Predistribution

Assume that there are N nodes in a wireless sensor network, where $2^{n-1} < N \le 2^n$. A n-dimension *H*2(*n*) is then generated and we construct a polynomial pool with the following method:

1) The key setup server randomly generates $n*2^n$ bivariate *t*-degree polynomial pool over a finite fields F_q ,denoted as $F = \{ f^i_{<i_1,i_2,...,i_{\lfloor n/2 \rfloor -1}>}(x,y),\ f^j_{<j_1,j_2,...,j_{\lceil n/2 \rceil -1}>}(x,y) \mid 0 \le i_1 \le i_2 \le ... \le i_{\lfloor n/2 \rfloor -1} \le 1, 1 \le i \le \lceil n/2 \rceil, 0 \le j_1 \le j_2 \le ... \le j_{\lceil n/2 \rceil -1} \le 1, 1 \le j \le \lceil n/2 \rceil \}$.

2) The $2^{\lceil n/2 \rceil -1}$ bivariate polynomials, denoted as $\{ f^j_{<j_1,j_2,...,j_{\lceil n/2 \rceil -1}>}(x,y) \mid 0 \le j_1 \le j_2 \le ... \le j_{\lceil n/2 \rceil} \le 1\}$, where $1 \le j \le \lceil n/2 \rceil$, are assigned to the *jth* dimension of the $(i_1, i_2, ..., i_{\lfloor n/2 \rfloor})$ th hypercube in *H*2(*n*).

3) The $2^{\lceil n/2 \rceil -1}$ bivariate polynomials, denoted as$\{ f^i_{<i_1,i_2,...,i_{\lfloor n/2 \rfloor -1}>}(x,y) \mid 0 \le i_1 \le i_2 \le ... \le i_{\lfloor n/2 \rfloor -1} \le 1\}$ where$1 \le i \le \lceil n/2 \rceil$, are assigned to the *ith* dimension of the $(j_1, j_2, ..., j_{\lceil n/2 \rceil})$ th hypercube in *H*2(*n*).

4) For any nodes $((i_1, i_2, ..., i_{\lfloor n/2 \rfloor}), (j_1, j_2, ..., j_{\lceil n/2 \rceil}))$ in *H*2(*n*), the polynomial shares, denoted as $\{ f^1_{<j_2,...,j_{\lceil n/2 \rceil}>}(x,y),\ f^2_{<j_1,j_3,...,j_{\lceil n/2 \rceil}>}(x,y), \ldots, f^{\lceil n/2 \rceil}_{<j_1,j_2,...,j_{\lceil n/2 \rceil -1}>}(x,y)\} \cup \{ f^1_{<i_2,i_3,...,i_{\lfloor n/2 \rfloor}>}(x,y),\ f^2_{<i_1,i_3,...,i_{\lfloor n/2 \rfloor}>}$

(x,y),…, $f^{\lfloor n/2\rfloor}_{<i_1,i_2,\dots,i_{\lfloor n/2\rfloor-1}>}$ (x,y)}, are assigned and pre-loaded before deployment phase.

5) The server assigns a unique ID, denoted as $((i_1,i_2,...,i_{\lfloor n/2\rfloor})$, $(j_1,j_2,...,j_{\lceil n/2\rceil})$, to every node in sequence, where $0 \le i_1 \le i_2 \le ... \le i_{\lfloor n/2\rfloor} \le 1$, $0 \le j_1 \le j_2 \le ... \le j_{\lceil n/2\rceil} \le 1$.

4.2. Direct Key Establishment

If any two nodes $A((i_1,i_2,...,i_{\lfloor n/2\rfloor}),(j_1,j_2,...,j_{\lceil n/2\rceil}))$ and $B((i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}), (j'_1,j'_2,...,j'_{\lceil n/2\rceil}))$ want to establish pairwise key, the node A can achieve the pairwise key with B by processing the following procedures:

Node *A* first computes the *Hamming* distance between B and itself, as $d_1=d_h((i_1,i_2,...,i_{\lfloor n/2\rfloor}), (i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}))$, $d_2=d_h((j_1,j_2,...,j_{\lceil n/2\rceil}), (j'_1,j'_2,...,j'_{\lceil n/2\rceil}))$. If $d_1=1$ or $d_2=1$, the node can establish the pairwise with the peer according to the conclusion of the Theorem 4.

Theorem 4: For any two nodes $A((i_1,i_2,...,i_{\lfloor n/2\rfloor}), (j_1,j_2,...,j_{\lceil n/2\rceil}))$ and $B((i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}), (j'_1,j'_2,...,j'_{\lceil n/2\rceil}))$, If the *Hamming* distance $d_h((i_1,i_2,...,i_{\lfloor n/2\rfloor}), (i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}))=1$, or $d_h((j_1,j_2,...,j_{\lceil n/2\rceil}), (j'_1,j'_2,...,j'_{\lceil n/2\rceil}))=1$, then there exists certainly pairwise key between *A* and *B*.

Proof: 1) $d_h((i_1,i_2,...,i_{\lfloor n/2\rfloor}),(i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}))=1$: Assume that $i_t=i'_t$, where $1 \le t \le \lfloor n/2\rfloor -1$. Since $i_{\lfloor n/2\rfloor} \ne i'_{\lfloor n/2\rfloor} \Rightarrow f^{\lfloor n/2\rfloor}_{<i_1,i_2,\dots,i_{\lfloor n/2\rfloor-1}>}(i_{\lfloor n/2\rfloor}, i'_{\lfloor n/2\rfloor}) = f^{\lfloor n/2\rfloor}_{<i'_1,i'_2,\dots,i'_{\lfloor n/2\rfloor-1}>}(i'_{\lfloor n/2\rfloor}, i_{\lfloor n/2\rfloor})$. So, There exists a pairwise key $f^{\lfloor n/2\rfloor}_{<i_1,i_2,\dots,i_{\lfloor n/2\rfloor-1}>}(i_{\lfloor n/2\rfloor}, i'_{\lfloor n/2\rfloor})$ between *A* and *B*.

2) $d_h((j_1,j_2,...,j_{\lceil n/2\rceil}), (j'_1,j'_2,...,j'_{\lceil n/2\rceil}))=1$: Imitating the step 1), it is easy to prove that there exists a pairwise key between *A* and *B*.

4.3. Indirect Key Establishment

If $d_h((i_1,i_2,...,i_{\lfloor n/2\rfloor}), (i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}))>1$ and $d_h((j_1,j_2,...,j_{\lceil n/2\rceil}), (j'_1,j'_2,...,j'_{\lceil n/2\rceil}))>1$ then node *A* establish indirect pairwise key with B according to Theorem 5. In order to make it clear, we will provide a lemma before the illustration of Theorem 5.

Lemma1: For any two nodes $A((i_1,i_2,...,i_{\lfloor n/2\rfloor}), (j_1,j_2,...,j_{\lceil n/2\rceil}))$ and $B((i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}), (j'_1,j'_2,...,j'_{\lceil n/2\rceil}))$, assume that $d_h=k$, then there exists a *k*-distance path denoted as $I_0(=A)$, $I_1,\dots,I_{k-1}$, $I_k(=B)$, where $d_h(I_i,I_j)=1$.

Proof: According to Theorem 3, d_h can be expressed as $d_h((i_1,i_2,...,i_{\lfloor n/2\rfloor}), (i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}))+d_h((j_1,j_2,...,j_{\lceil n/2\rceil}), (j'_1,j'_2,...,j'_{\lceil n/2\rceil}))=k$. Assume that $d_h((i_1,i_2,...,i_{\lfloor n/2\rfloor}), (i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}))=h$, then $d_h((j_1,j_2,...,j_{\lceil n/2\rceil}), (j'_1,j'_2,...,j'_{\lceil n/2\rceil}))=k-h$. According to definition5, node $C((i_1,i_2,...,i_{\lfloor n/2\rfloor}), (j'_1,j'_2,...,j'_{\lceil n/2\rceil}))$ and *A* are located in a $\lceil n/2\rceil$ -dimensional hypercube *H*, and node *C* and B are located in a $\lceil n/2\rceil$-dimensional hypercube H'. According to the properties of hypercube [5,6], there exist a path described as $I_0(=A)$, $I_1,\dots, I_{h-1}$, $I_h(=C)$ in *H*, where $d_h(I_i, I_j)=1$. Similarly, another path with the same property is existed in H', denoted as $I_h(=A)$, $I_{h+1}, \dots, I_{k-1}$, $I_k(=B)$, where $d_h(I_i,I_j)=1$.

Thus there exist a integrated path in *H*2 diagram from node *A* to *B*, described as $I_0(=A)$, $I_1,\dots,I_{k-1}$, $I_k(=B)$ where $d_h(I_i, I_j)=1$.

Theorem 5: Assume that any two nodes can communicate directly in a wireless sensor networks, and there is no compromised node in the networks, then there exist a key path for any node $A((i_1,i_2,...,i_{\lfloor n/2\rfloor}), (j_1,j_2,...,j_{\lceil n/2\rceil}))$ and node $B((i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}), (j'_1,j'_2,...,j'_{\lceil n/2\rceil}))$.

Proof: According to Lemma1, there exist a path for any two nodes where $d_h=k$ in *H*2 diagram. Thus the conclusion is held.

We propose the algorithm for indirect key establishment as follows. Assume the two nodes $A((i_1,i_2,...,i_{\lfloor n/2\rfloor}), (j_1,j_2,...,j_{\lceil n/2\rceil}))$ and node $B((i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}), (j'_1,j'_2,...,j'_{\lceil n/2\rceil}))$ want to establish indirect pairwise key in the network, we propose the algorithm for indirect key establishment illustrated as follows.

Indirect_Key_Establishing_Algorithm(){

1) Node A computes a set *L* which records the dimensions in which node A and B have different sub-indexes. The set can be expressed as $L=\{(d_1,d_2,\dots,d_k),(g_1,g_2,\dots,g_w)\}$ where $d_1< d_2<\dots< d_k$, $g_1<g_2<\dots<g_w$.

2) Node A maintains a path set *P* with initial vale of $P=\{A\}$.

3) Assume that $U((u_1,u_2,...,u_{\lfloor n/2\rfloor}),(u'_1,u'_2,...,u'_{\lceil n/2\rceil}))=A$; $s=1$.

4) Node *A* computes intermediate nodes expressed as $V=((u_1,u_2,\dots u_{d_s-1}, i'_{d_s}, u_{d_s+1},\dots,u_{\lfloor n/2\rfloor}),(u'_1,u'_2,...,u'_{\lceil n/2\rceil}))$. And $P=P\cup\{V\}$.

5) Assume that $U = V$.

6) If $s<k$, then $s=s+1$, and repeats the step 4, otherwise turns to step7).

7) Node A computes intermediate nodes V= ($(u_1, u_2, ..., u_{\lfloor n/2 \rfloor})$, $(u'_1, u'_2, ... u'_{g_s-1}, j'_{g_s}, u'_{g_s+1}, ..., u'_{\lceil n/2 \rceil})$), and let $P=P \cup \{V\}$.

8) Let $U = V$.

9) If $s<w$, then $s=s+1$, and repeats step7); otherwise go on step10).

10) Let $P=P \cup \{B\}$.

}

According to Theorem 5, any node can compute a key path to it destination when there is no compromised node in the network. Once the path P is achieved, the two nodes can exchange secret information to generate pairwise key between themselves.

For example, the node A((001), (0101)) and the node B((100), (1100)) can establish pairwise key along the following key path: A((001), (0101)) $\rightarrow$ ((101), (0101)) $\rightarrow$ ((100), (0101)) $\rightarrow$ ((100), (1101)) $\rightarrow$ B((100), (1100)).

According to the algorithm described above, the following conclusion is naturally held.

Theorem 6: Assume that any two nodes can communicate with each other directly, and there is no compromised node in a network. If the distance between the two nodes is k, then there exists a key path with distance of k. That is, the two nodes can establish pairwise key through k-1 intermediate nodes.

4.4. Dynamic Key Path Establishment

The Indirect_Key_Establishing_Algorithm() illustrated in Subsection 4.3 can only deal with the situation that there is no compromised node in the network. However, in case of some existent compromised nodes, the algorithm would fail to find fungible intermediate node to help establish pairwise key.

We further analyze the example addressed in Subsection 4.3. When the node ((101),(0101)) is compromised, the node A and B can utilize the following path to establish pairwise key: A((001), (0101)) $\rightarrow$ ((000),(0101)) $\rightarrow$ ((100), (0101)) $\rightarrow$ ((100), (1101)) $\rightarrow$ B((100), (1100)).

When the node ((100),(1101)) is compromised, the two nodes can use the path: A((001), (0101)) $\rightarrow$ ((101), (0101)) $\rightarrow$ ((100), (0101)) $\rightarrow$ ((100), (0100)) $\rightarrow$ B((100), (1100)).

In case that the nodes ((101),(0101)),((100),(1101)) are compromised, there still exists a key path denoted as A((001),(0101)) $\rightarrow$ ((000),(0101)) $\rightarrow$ ((100),(0101)) $\rightarrow$ ((100),(0100)) $\rightarrow$ B((100),(1100)).

4.4.1. Relative Definitions of Local Weak Connectivity

Definition 6: The nodes A and B in a n-dimensional hypercube H_n are called neighbors, if that there exists only one different bit in their binary strings.

Definition 7: The node A in an m-dimensional hypercube/sub-hypercube H_m is m-disconnected, iif that all links between A and every faultless node in H_m are fault. The node A in an m-dimensional hypercube/sub- hypercube H_m is reachable, iif that A is faultless and not m-disconnected.

Definition 8 (k-dimensional local-weak-connectivity): A n-dimensional hypercube H_n is k-dimensional local-weak-connected, if all reachable nodes in each k-dimensional sub-hypercube H_k ($k \geq 1$) of H_n forms a connected graph, and the number of reachable nodes in H_k is bigger than 2^{k-1}.

Definition 9 (general local-weak-connectivity): An n-dimensional hypercube H_n is general local-weak- connected, if there exists a h-dimensional sub-hypercube H_h ($h \geq k$), which is local-weak-connected and includes H_k, as for each k-dimensional sub-hypercube H_k ($k \geq 1$) of H_n.

Figure 2 presents a 3-dimensional hypercube H_3 with two fault nodes and a 3-disconnected node. According to the above two kinds of local-weak-connectivity concepts, it is easy to prove that all reachable nodes in H_3 is global connected.

4.4.2. Global Connectivity of Local-Weak-Connected Hypercube

An n-dimensional hypercube H_n has 2^n nodes, in which each node can be represented by a binary string and has n different links. So H_n has n 2^{n-1} different links totally.

Definition 10: Any binary string $b_1\, b_2 \cdots b_{n-k}$ with given length n-k corresponds a k-dimensional sub-hypercube H_k with 2^k nodes, and the nodes in H_k can be represented by such binary strings as $b_1\, b_2 \cdots b_{n-k}$ *…*, where * can be 0 or 1.

From the construction algorithm of H_k, it is easy to know that all k-dimensional sub-hypercubes are isomor-

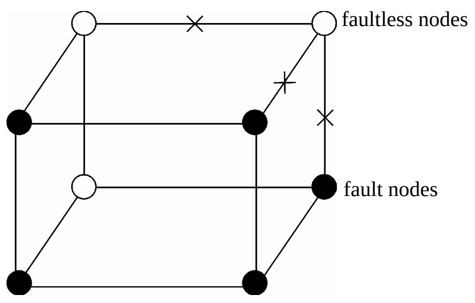

Figure 2. A 3-dimensional hypercube H3 with fault nodes and links. In which, the black dots represent fault nodes, the white dots represent faultless nodes, the lines with tokens represent fault links, and the lines without tokens represent faultless links.

phic, and H_n includes all of the k-dimensional sub-hypercubes that is isomorphic with H_k. And it is easy to prove that H_n includes 2^{n-k} -1 k-dimensional sub-hypercubes that is isomorphic with H_k and has no common nodes with H_k also.

Lemma 2: All of the reachable nodes in any two neighboring k-dimensional sub-hypercubes of the Local-Weak-Connected n-dimensional Hypercube H_n form a connected graph.

Proof: Let that H_k and H'_k are two neighboring k-dimensional sub-hypercubes, according to definition 10, we can utilize binary strings $b_1\, b_2 \cdots b_{n-k}$ *…* to represent the nodes in H_k, where * can be 0 or 1. Since H_k and H'_k are neighboring, so there exists at least one common node between H_k and $H'_k \Rightarrow$ the nodes in H'_k can be represented as $b_1 \cdots b_{t-1}$ * $b_{t+1} \cdots b_{n-k}$ *…* d_s *…* and $b_1 \cdots b_{t-1}\ b_t\ b_{t+1} \cdots b_{n-k}$ *…* d_s *…* $\in H_k$. $\Rightarrow$ $b_1 \cdots b_{t-1}\ b_t\ b_{t+1} \cdots b_{n-k}$ *…* d_s *…* $\in H_k \cap H'_k$. Considering that the number of reachable nodes in H'_k is bigger than $2^{k-1} \Rightarrow$ there exists at least one node A in those 2^{k-1} nodes represented by $b_1 \cdots b_{t-1}\ b_t\ b_{t+1} \cdots b_{n-k}$ *… * d_s *…* is reachable. Since $A \in H_n \Rightarrow$ Node A and all of the reachable nodes in H_k form a connected graph. And in addition, since $A \in H'_k \Rightarrow$ Node A and all of the reachable nodes in H'_k form a connected graph also. So, all of the reachable nodes in H_k and H'_k form a connected graph.

Theorem 7: All of the reachable nodes in n-dimensional Hypercube H_n, which satisfies the conditions of k-dimensional local-weak-connectivity, form a connected graph.

Proof: From Theorem 2, it is easy to prove that the conclusion stands.

From Theorem 7, we can obtain the following deduces easily.

Deduce 1: If n-dimensional Hypercube H_n satisfies the conditions of k-dimensional local-weak-connectivity, then all of the reachable nodes in any h-dimensional sub-hypercube $H_h(h \geq k)$ form a connected graph.

Deduce 2: If n-dimensional Hypercube H_n satisfies the conditions of k-dimensional local-weak-connectivity, then there exists at least a pair of connected reachable nodes $a_1 \cdots a_{j-1}\ 0\ a_{j+1} \cdots a_{n-k}\ \chi_{n-k+1} \cdots \chi_n$ and $a_1 \cdots a_{j-1}\ 1\ a_{j+1} \cdots a_{n-k}\ \chi'_{n-k+1} \cdots \chi'_n$ between any pair of k-dimensional sub-hypercubes of $a_1 \cdots a_{j-1}\ 0\ a_{j+1} \cdots a_{n-k}$ *…* and $a_1 \cdots a_{j-1}\ 1\ a_{j+1} \cdots a_{n-k}$ *…*.

Proof: From deduce 1, if n-dimensional Hypercube H_n satisfies the conditions of k-dimensional local-weak-connectivity $\Rightarrow$ All of the reachable nodes in any h-dimensional sub-hypercube $H_h(h \geq k)$ form a connected graph $\Rightarrow$ All of the reachable nodes in any $(k+1)$-dimensional sub-hypercube $a_1 \cdots a_{j-1}$ * $a_{j+1} \cdots a_{n-k}$ *…* form a connected graph. And since the numbers of unreachable nodes in k-dimensional sub-hypercubes $a_1 \cdots a_{j-1}\ 0\ a_{j+1} \cdots a_{n-k}$ *…* and $a_1 \cdots a_{j-1}\ 1\ a_{j+1} \cdots a_{n-k}$ *…* are less than a half of the number of total nodes respectively. So, there exists reachable node $a_1 \cdots a_{j-1}\ 0\ a_{j+1} \cdots a_{n-k}\ \chi_{n-k+1} \cdots \chi_n$ in $a_1 \cdots a_{j-1}\ 0\ a_{j+1} \cdots a_{n-k}$ *…*, and there exists reachable node $a_1 \cdots a_{j-1}\ 1\ a_{j+1} \cdots a_{n-k}\ \chi'_{n-k+1} \cdots \chi'_n$ in $a_1 \cdots a_{j-1}\ 1\ a_{j+1} \cdots a_{n-k}$ *…* certainly. And in addition, those two nodes $a_1 \cdots a_{j-1}\ 0\ a_{j+1} \cdots a_{n-k}\ \chi_{n-k+1} \cdots \chi_n$ and $a_1 \cdots a_{j-1}\ 1\ a_{j+1} \cdots a_{n-k}\ \chi'_{n-k+1} \cdots \chi'_n$ is connected.

Theorem 8: All of the reachable nodes in n-dimensional Hypercube H_n, which satisfies the conditions of general local-weak-connectivity, form a connected graph.

Proof: Since n-dimensional Hypercube H_n is local-weak-connected, and there exists no other sub-hypercubes that include itself in H_n. So, from definition 9, it is easy to know that H_n is n-dimensional local-weak-connected. And in addition, from definition 8, we can know that all of the reachable nodes in H_n form a connected graph.

Theorem 7 and Theorem 8 show that the hypercubes, that satisfy the conditions of the proposed two kinds of local-weak-connectivity, must be global connected.

4.4.3. K-Dimensional Local-Weak-Connectivity Based Dynamic Key Path Establishment Algorithm

KLWC-based Dynamic_Key_Path_Establishing_Algorithm(){

Input: Sensor network H_n with fault nodes and fault links (The links, whose length are bigger than the transmitting radius). And two reachable nodes $A((i_1, i_2, ..., i_{\lfloor n/2 \rfloor}) , (j_1, j_2, ..., j_{\lceil n/2 \rceil}))$ and $B((i'_1, i'_2, ..., i'_{\lfloor n/2 \rfloor}) , (j'_1, j'_2, ..., j'_{\lceil n/2 \rceil}))$ in H_n.

Output: A correct key path P from A to B in H_n.

1) Compute and determine the node T=$((i'_1, i'_2, ..., i'_{\lfloor n/2 \rfloor}) , (j_1, j_2, ..., j_{\lceil n/2 \rceil}))$ in $H_{\lfloor n/2 \rfloor}$;

2) P=Dynamic_Key_Path_Establishment_1 (A,T);

3) P=P $\cup$ Dynamic_Key_Path_Establishment_2 (T, B);

4) If P is a correct key path from A to B, then exit, otherwise turn to step 5);

5) Compute and determine the node T=$((i_1, i_2, ..., i_{\lfloor n/2 \rfloor}) , (j'_1, j'_2, ..., j'_{\lceil n/2 \rceil}))$ in $H_{\lfloor n/2 \rfloor}$;

6) P=Dynamic_Key_Path_Establishment_2 (A,T);

7) $P=P \cup$ Dynamic_Key_Path_Establishment_1 (T, B);

8) If P is a correct key path from A to B, then exit, otherwise turn to step 9);

9) Report A, failure to establish a key path from A to B.

}

Algorithm Dynamic_Key_Path_Establishment_1(*A*,*T*):

1) Obtain the codes of nodes A and T: $A \leftarrow ((i_1, i_2, ..., i_{\lfloor n/2 \rfloor}), (j_1, j_2, ..., j_{\lceil n/2 \rceil}))$, $T \leftarrow ((i'_1, i'_2, ..., i'_{\lfloor n/2 \rfloor}), (j_1, j_2, ..., j_{\lceil n/2 \rceil}))$;

/* From definition 10, we can suppose that the k-dimensional sub-hypercube that includes T is $((i'_1\ i'_2 \cdots i'_{\lfloor n/2 \rfloor - k} * \ldots *), (j_1, j_2, ..., j_{\lceil n/2 \rceil}))$.*/

2) Initialize Path P: $P \leftarrow A$;

3) Initialize temporary binary string C: $C=((c_1, c_2, ..., c_{\lfloor n/2 \rfloor}), (j_1, j_2, ..., j_{\lceil n/2 \rceil})) \leftarrow A$;

4) FOR(j=1; $j \le \lfloor n/2 \rfloor$; j++){

IF($i_j \neq i'_j$){

①According to lemma2 and deduce 1, a pair of connected reachable nodes C and D can be found through discovering in neighboring k-dimensional sub- hypercubes:

$$C=((c_1, c_2, \ldots c_{j-1}, c_j, c_{j+1}, ..., c_{\lfloor n/2 \rfloor}), (j_1, j_2, ..., j_{\lceil n/2 \rceil})),$$

$$D=((i'_1, i'_2, \ldots i'_{j-1}, i'_j, x_{j+1}, ..., x_{\lfloor n/2 \rfloor}), (j_1, j_2, ..., j_{\lceil n/2 \rceil})),$$

where $c_t = i'_t$ ($t \in [1, j]$);

②Join the path from $((i'_1, i'_2, \ldots i'_{j-1}, i_j, i_{j+1}, ..., i_{\lfloor n/2 \rfloor}), (j_1, j_2, ..., j_{\lceil n/2 \rceil}))$ to node D into P;

③$C \leftarrow D$;

}

}

/* After the above steps, a correct key path from node $A((i_1, i_2, ..., i_{\lfloor n/2 \rfloor}), (j_1, j_2, ..., j_{\lceil n/2 \rceil}))$ to the reachable node $((i'_1\ i'_2 \cdots i'_{\lfloor n/2 \rfloor - k}\ \chi'_{\lfloor n/2 \rfloor - k+1} \cdots \chi'_{\lfloor n/2 \rfloor}), (j_1, j_2, ..., j_{\lceil n/2 \rceil}))$ will be constructed. */

5) Join the path from node $((i'_1 i'_2 \cdots i'_{\lfloor n/2 \rfloor - k}\ \chi'_{\lfloor n/2 \rfloor - k+1} \cdots \chi'_{\lfloor n/2 \rfloor}), (j_1, j_2, ..., j_{\lceil n/2 \rceil}))$ to node T in the k-dimensional sub-hypercube $((i'_1\ i'_2 \cdots i'_{\lfloor n/2 \rfloor - k} **), (j_1, j_2, ..., j_{\lceil n/2 \rceil}))$ into P. And then exit. So, a correct key path from node A to node T is discovered.

Algorithm Dynamic_Key_Path_Establishment_2(*T*, *B*):

1) Obtain the codes of nodes B and T: $T \leftarrow ((i'_1, i'_2, ..., i'_{\lfloor n/2 \rfloor}), (j_1, j_2, ..., j_{\lceil n/2 \rceil}))$; $B \leftarrow ((i'_1, i'_2, ..., i'_{\lfloor n/2 \rfloor}), (j'_1, j'_2, ..., j'_{\lceil n/2 \rceil}))$;

/* From definition 10, we can suppose that the k-dimensional sub-hypercube that includes B is $((i'_1, i'_2, ..., i'_{\lfloor n/2 \rfloor}), (j'_1\ j'_2 \cdots j'_{\lceil n/2 \rceil - k} * \ldots *))$.*/

2) Initialize Path P: $P \leftarrow T$;

3) Initialize temporary binary string C: $C=((i'_1, i'_2, ..., i'_{\lfloor n/2 \rfloor}), (c_1, c_2, ..., c_{\lceil n/2 \rceil})) \leftarrow T$;

4) FOR(l=1; $l \le \lceil n/2 \rceil$; l++){

IF($j_l \neq j'_l$){

① According to lemma2 and deduce 1, a pair of connected reachable nodes C and D can be found through discovering in neighboring k-dimensional sub-hypercubes:

$$C=((i'_1, i'_2, ..., i'_{\lfloor n/2 \rfloor}), (c_1, c_2, \cdots c_{l-1}, c_l, c_{l+1}, ..., c_{\lceil n/2 \rceil}));$$

$$D=((i'_1, i'_2, ..., i'_{\lfloor n/2 \rfloor}), (j'_1, j'_2, \ldots j'_{l-1}, j'_l, x_{l+1}, ..., x_{\lceil n/2 \rceil})),$$

where $c_t = j'_t$ ($t \in [1, l]$).

② Join the path from $((i'_1, i'_2, ..., i'_{\lfloor n/2 \rfloor}), (j'_1, j'_2, \ldots j'_{l-1}, j_l, j_{l+1}, ..., j_{\lceil n/2 \rceil}))$ to node D into P;

③ $C \leftarrow D$;

}

}

/* After the above steps, a correct key path from node $T((i'_1, i'_2, ..., i'_{\lfloor n/2 \rfloor}), (j_1, j_2, ..., j_{\lceil n/2 \rceil}))$ to the reachable node $((i'_1, i'_2, ..., i'_{\lfloor n/2 \rfloor}), (j'_1\ j'_2 \cdots j'_{\lceil n/2 \rceil - k}\ \chi'_{\lceil n/2 \rceil - k+1} \cdots \chi'_{\lceil n/2 \rceil}))$ will be constructed. */

5) Join the path from node $((i'_1, i'_2, ..., i'_{\lfloor n/2 \rfloor}), (j'_1\ j'_2 \cdots j'_{\lceil n/2 \rceil - k}\ \chi'_{\lceil n/2 \rceil - k+1} \cdots \chi'_{\lceil n/2 \rceil}))$ to node B in the k-dimensional sub-hypercube $((i'_1, i'_2, ..., i'_{\lfloor n/2 \rfloor}), (j'_1\ j'_2 \cdots j'_{\lceil n/2 \rceil - k} **))$ into P. And then exit. So, a correct key path from node T to node B is discovered.

From the above description, we can know that the time complexity of algorithm Dynamic_Key_Path_ Establishment_1 is

$O((\lfloor n/2 \rfloor - k)2^k) + O(2^k) = O(n2^{k-1})$, and the time complexity of algorithm Dynamic_Key_Path_Establishment_2 is $O((\lceil n/2 \rceil - k)2^k) + O(2^k) = O(n2^{k-1})$, so the total time complexity of the k-Dimensional Local-Weak-Connectivity based Dynamic Key Path Establishment Algorithm is $O(n2^k)$.

Considering the percentage of the fault nodes in sensor networks, when applying the k-Dimensional Local-Weak-Connectivity based Dynamic Key Path Establishment Algorithm actually, we can set k=1,2,3. Then the total time complexity of the k-Dimensional Local-Weak-Connectivity based Dynamic Key Path Establishment Algorithm will be O（*n*） only. Figure 3 illustrates the relationship of dimension *n* and the scale of the sensor networks.

4.4.4. General Local-Weak-Connectivity Based Dynamic Key Path Establishment Algorithm

```
GLWC-based Dynamic_Key_Path_Establishing_Algorithm(){
    Input: Sensor network $H_n$ with fault nodes and fault links (The links, whose length are bigger than the transmitting radius). And two reachable nodes $A((i_1,i_2,...,i_{\lfloor n/2\rfloor}),(j_1,j_2,...,j_{\lceil n/2\rceil}))$ and $B((i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}),(j'_1,j'_2,...,j'_{\lceil n/2\rceil}))$ in $H_n$.
    Output: A correct key path $P$ from $A$ to $B$ in $H_n$.
    1) Compute and determine the node $T=((i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}),(j_1,j_2,...,j_{\lceil n/2\rceil}))$ in $H_{\lfloor n/2\rfloor}$;
    2)$P$=Dynamic_Key_Path_Establishment_3($A$, $T$);
    3)$P=P \cup$ Dynamic_Key_Path_Establishment_4($T$, $B$);
    4)If $P$ is a correct key path from $A$ to $B$, then exit, otherwise turn to step 5);
    5)Compute and determine the node $T=((i_1,i_2,...,i_{\lfloor n/2\rfloor}),(j'_1,j'_2,...,j'_{\lceil n/2\rceil}))$ in $H_{\lfloor n/2\rfloor}$;
    6)$P$=Dynamic_Key_Path_Establishment_4($A$, $T$);
    7)$P=P \cup$ Dynamic_Key_Path_Establishment_3($T$, $B$);
    8)If $P$ is a correct key path from $A$ to $B$, then exit, otherwise turn to step 9);
    9)Report $A$, failure to establish a key path from $A$ to $B$.
}
```

Algorithm Dynamic_Key_Path_Establishment_3(*A*, *T*):

```
1)Obtain the codes of nodes $A$ and $T$: $A \leftarrow ((i_1,i_2,...,i_{\lfloor n/2\rfloor}),(j_1,j_2,...,j_{\lceil n/2\rceil}))$, $T \leftarrow ((i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}),(j_1,j_2,...,j_{\lceil n/2\rceil}))$;
/* From definition 10, we can suppose that the $k$-dimensional sub-hypercube that includes $T$ is $((i'_1\ i'_2 \cdots i'_{\lfloor n/2\rfloor-k} *\ldots*),(j_1,j_2,...,j_{\lceil n/2\rceil}))$.*/
2)Initialize Path $P$: $P \leftarrow A$;
3)Initialize temporary binary string $C$: $C=((c_1,c_2,...,c_{\lfloor n/2\rfloor}),(j_1,j_2,...,j_{\lceil n/2\rceil})) \leftarrow A$;
4)FOR($j$=1; $j \le n$; $j$++){
    IF($i_j \ne i'_j$){
    FOR($k$=1; $k \le n-j$; $k$++){
    IF(According to Theorem 8, a pair of connected reachable nodes $C$ and $D$ can be found through discovering in neighboring $k$-dimensional sub-hypercubes:
    $C=((c_1,c_2,...c_{j-1},c_j,c_{j+1},...,c_{\lfloor n/2\rfloor}),(j_1,j_2,...,j_{\lceil n/2\rceil}))$,
    $D=((i'_1,i'_2,...i'_{j-1},i'_j,x_{j+1},...,x_{\lfloor n/2\rfloor}),(j_1,j_2,...,j_{\lceil n/2\rceil}))$, where $c_t=i'_t$ ($t\in$ [1,$j$]);
    ①Join the path from $((i'_1,i'_2,...i'_{j-1},i_j,i_{j+1},...,i_{\lfloor n/2\rfloor}),(j_1,j_2,...,j_{\lceil n/2\rceil}))$ to node $((i'_1,i'_2,...i'_{j-1},i'_j,x_{j+1},...,x_{\lfloor n/2\rfloor}),(j_1,j_2,...,j_{\lceil n/2\rceil}))$ into $P$;
    ②$C \leftarrow ((i'_1,i'_2,...i'_{j-1},i'_j,x_{j+1},...,x_{\lfloor n/2\rfloor}),(j_1,j_2,...,j_{\lceil n/2\rceil}))$;
    ③Break;
        }
    }
    IF($k > n-j$){
    WHILE($k \le n$){
    IF(In the $k$-dimensional hypercube $((c_1,c_2,...c_{\lfloor n/2\rfloor-k}*...*),(j_1,j_2,...,j_{\lceil n/2\rceil}))$, there exists no faultless key path from node $C$ to node $T$) $k$++;
        }
    }
    IF( $k > n$) exit. Then $H_{\lfloor n/2\rfloor}$ is not general local-weak-connected, and we cannot find a correct key path from node $A$ to $T$.
    ELSE Join the path from $C$ to $T$ in the $k$-dimensional hypercube $((c_1,c_2,...c_{\lfloor n/2\rfloor-k}*...*),(j_1,j_2,...,j_{\lceil n/2\rceil}))$ into $P$;
        }
    }
5) Exit. And a correct key path from $A$ to $T$ is discovered.
```

Algorithm Dynamic_Key_Path_Establishment_4(*T*, *B*):

```
1) Obtain the codes of nodes $T$ and $B$: $T \leftarrow ((i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}),(j_1,j_2,...,j_{\lceil n/2\rceil}))$ and $B \leftarrow ((i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}),(j'_1,j'_2,...,j'_{\lceil n/2\rceil}))$;
/* From definition 10, we can suppose that the $k$-dimensional sub-hypercube that includes $B$ is $((i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}),(j'_1\ j'_2 \cdots j'_{\lceil n/2\rceil-k} *\ldots*))$.*/
2) Initialize Path $P$: $P \leftarrow T$;
3) Initialize temporary binary string $C$: $C=((i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}),(c_1,c_2,...,c_{\lceil n/2\rceil})) \leftarrow T$;
4) FOR($l$=1; $l \le n$; $l$++){
    IF( $j_l \ne j'_l$ ){
    FOR($k$=1; $k \le n-j$; $k$++){
    IF(According to Theorem 8, a pair of connected reachable nodes $C$ and $D$ can be found through discovering in neighboring $k$-dimensional sub-hypercubes:
    $C=((i'_1,i'_2,...,i'_{\lfloor n/2\rfloor}),(c_1,c_2,...c_{j-1},c_j,c_{j+1},...,c_{\lceil n/2\rceil}))$,
        $D$=( $(i'_1, i'_2, \dots, i'_{\lfloor n/2 \rfloor})$ ,
        $(j'_1, j'_2, \dots j'_{l-1}, j'_l, x_{l+1}, \dots, x_{\lceil n/2 \rceil})$ ),
        where $c_t = i'_t$ $(t \in [1,j])$;
    ①Join the path from ( $(i'_1, i'_2, \dots, i'_{\lfloor n/2 \rfloor})$ , $(j'_1, j'_2, \dots j'_{l-1}, j_l, j_{l+1}, \dots, j_{\lceil n/2 \rceil})$ ) to node ( $(i'_1, i'_2, \dots, i'_{\lfloor n/2 \rfloor})$ , $(j'_1, j'_2, \dots j'_{l-1}, j'_l, x_{l+1}, \dots, x_{\lceil n/2 \rceil})$ ) into $P$;
    ②$C \leftarrow$ ( $(i'_1, i'_2, \dots, i'_{\lfloor n/2 \rfloor})$ , $(j'_1, j'_2, \dots j'_{l-1}, j'_l, x_{l+1}, \dots, x_{\lceil n/2 \rceil})$ );
    ③Break;
      }
    }
    IF($k > n-j$){
    WHILE($k \le n$){
    IF(In the $k$-dimensional hypercube ( $(i'_1, i'_2, \dots, i'_{\lfloor n/2 \rfloor})$ , $(c_1, c_2, \dots c_{\lceil n/2 \rceil - k} * \dots *)$ ), there exists no faultless key path from node $C$ to node $B$) $k$++;
      }
    }
    IF($k > n$) exit. Then $H_{\lceil n/2 \rceil}$ is not general local-weak-connected, and we cannot find a correct key path from node $T$ to $B$.
    ELSE Join the path from $C$ to $B$ in the $k$-dimensional hypercube ( $(i'_1, i'_2, \dots, i'_{\lfloor n/2 \rfloor})$ , $(c_1, c_2, \dots c_{\lceil n/2 \rceil - k} * \dots *)$ ) into $P$;
      }
    }
    5) Exit. And a correct key path from $T$ to $B$ is discovered.
```

From the above description, we can know that the time complexity of algorithm Dynamic_Key_Path_ Establishment_3 is $O(\lfloor n/2 \rfloor 2^{k_{\min}}) + O(2^{k_{\min}}) = O(\lfloor n/2 \rfloor 2^{k_{\min}})$, where k_{min} is the smallest integer that satisfies the condition of k-dimensional local-weak-connectivity. And the time complexity of algorithm Dynamic_Key_Path_Establishment_4 is $O(\lceil n/2 \rceil 2^{k_{\min}}) + O(2^{k_{\min}}) = O(\lceil n/2 \rceil 2^{k_{\min}})$, so the total time complexity of the general Local-Weak- Connectivity based Dynamic Key Path Establishment Algorithm is $O(\lceil n/2 \rceil 2^{k_{\min}})$.

Considering the percentage of the fault nodes in sensor networks, when applying the general Local- Weak-Connectivity based Dynamic Key Path Establishment Algorithm actually, we can set k=1,2,3. Then the total time complexity of the General Local-Weak- Connectivity based Dynamic Key Path Establishment Algorithm will be $O(n)$ only.

5. Analysis

5.1. Feasibilities of the Algorithm

Theorem 9: In our algorithm, the possibility of direct key establishment for any two nodes can be expressed as $P_{H2} \approx (2^{\lfloor n/2 \rfloor} + 2^{\lceil n/2 \rceil})/(N-1)$.

Proof: As the algorithm has assigned any node, denoted as ($(i_1, i_2, \dots, i_{\lfloor n/2 \rfloor})$, $(j_1, j_2, \dots, j_{\lceil n/2 \rceil})$), shares of polynomials expressed as F_A = { $f^1_{<j_2,\dots,j_{\lceil n/2 \rceil}>}(j_1,y)$, $f^2_{<j_1,j_3,\dots,j_{\lceil n/2 \rceil}>}(j_2,y)$,…, $f^{\lceil n/2 \rceil}_{<j_1,j_2,\dots,j_{\lceil n/2 \rceil -1}>}(j_{\lceil n/2 \rceil},y)$} $\cup$ { $f^1_{<i_2,\dots,i_{\lfloor n/2 \rfloor}>}(i_1,y)$, $f^2_{<i_1,i_3,\dots,i_{\lfloor n/2 \rfloor}>}(i_2,y)$,…, $f^{\lfloor n/2 \rfloor}_{<i_1,i_2,\dots,i_{\lfloor n/2 \rfloor -1}>}(i_{\lfloor n/2 \rfloor},y)$}. It's clear that there are $2^{\lfloor n/2 \rfloor} + 2^{\lceil n/2 \rceil}$ nodes which can establish direct pairwise key with the node A. Thus $P_{H2} \approx (2^{\lfloor n/2 \rfloor} + 2^{\lceil n/2 \rceil})/(N-1)$ as the network scale is within the area $2^{n-1} < N \le 2^n$.

Suppose that a sensor network has N=10000 sensor nodes, then n=14. The possibility of direct key establish is about $P_{H2} \approx 2.56\%$ according to the conclusion drawn by Theorem 6. However, the possibility decreases to $P_H \approx 0.14\%$ if the algorithm addressed in [3] is used.

Theroem10: Assume that the possibility of direct key establishment in $H2$-based scheme is defined as P_{H2}, while the possibility in hypercube is denoted as P_H, then $P_{H2} >> P_H$.

Proof: Suppose the number of a network is within the area of $2^{n-1} < N \le 2^n$, and $P_H \approx \frac{n}{N-1}$ as addressed in [3]. Thus $\lim_{n\to\infty} \frac{P_H}{P_{H2}} = \lim_{n\to\infty} \frac{n}{2^{\lfloor n/2 \rfloor} + 2^{\lceil n/2 \rceil}} = 0$.

5.2. Overhead Analysis

Node's Storage Overhead

1) Any node is required to store t-degree bivariate polynomials whose number is n over the finite fields q, which occupies $n(t+1)\log q$ bits.

2) In order to keep the security of the Keys, for any bivariate polynomial $f(x,y)$, node A is required to store the ID information of the compromised nodes that can establish direct key with A by using $f(x,y)$. Since the degree of $f(x,y)$ is t, then $f(x,y)$ will be divulged when there are more than t nodes are compromised. So, for any bivariate polynomial $f(x,y)$, node A needs only to store the ID information of n compromised nodes that can establish direct key with A by using $f(x,y)$. In addition, since the node's ID is a vector of n bits, and from Theorem 4, we can know that node A needs only to store one bit for each compromised node to determine the whole ID information of the compromised node. So, the total storage cost is *nt bits*.

3) Also the node's own ID information occupies about *n bits* storage space, as it is expressed as $((i_1,i_2,\ldots,i_{\lfloor n/2\rfloor}),(j_1,j_2,\ldots,j_{\lceil n/2\rceil}))$.

All of the storage overhead address above sum up to $n(t+1)\log q+nt+n= n(t+1)\log 2q$ *bits*.

Theorem 11: The *H*2-based and the hypercube-based schemes have the same storage overhead.

Proof: According to the analysis on storage overhead addressed in Subsection 5.4 in [3], the result is certainly held.

Communication Overhead

In a sensor network, sending a unicast message between two arbitrary nodes may involve the overhead of establishing a route. In case of no compromised node existent in the network, any one node can communicate with the others directly. Assume that the overhead for a hop is defined as 1, then for two arbitrary nodes whose Hamming distance is *L*, the minimum communication overhead is *L*. We further inspect average communication overhead on *H*2-based path key establishment.

Suppose there are two nodes $A((i_1,i_2,\ldots,i_{\lfloor n/2\rfloor}),(j_1,j_2,\ldots,j_{\lceil n/2\rceil}))$ and $B((i'_1,i'_2,\ldots,i'_{\lfloor n/2\rfloor}),(j'_1,j'_2,\ldots,j'_{\lceil n/2\rceil}))$ In the formal part of node's code, the probability of $i_e=i'_e$, $e\in\{1,\ldots,\lceil n/2\rceil\}$ is 1/2; Similarly, the probability of $j_e=j'_e$, $e\in\{1,\ldots,\lceil n/2\rceil\}$ is also 1/2 in the latter code part. Thus the probability for the two nodes to have *i* different sub-index in the formal part is expressed as *P*[*i different sub-indexs in former part*]= $\frac{1}{2^{\lfloor n/2\rfloor}}\frac{(\lfloor n/2\rfloor)!}{i!(\lfloor n/2\rfloor-i)!}$. In the latter part, we also have:

P[*j different sub-indexs in later part*]= $\frac{1}{2^{\lceil n/2\rceil}}\frac{(\lceil n/2\rceil)!}{j!(\lceil n/2\rceil-j)!}$.

Thus the average communication overhead can be summarized as:

$$L=\sum_{i=1}^{\lfloor n/2\rfloor}(i-1)\times P[\text{i different sub-indexs in former part}]$$

$$+\sum_{j=1}^{\lceil n/2\rceil}(j-1)\times P[\text{j different sub-indexs in former part}].$$

Theroem12: The average communication overhead in the *H*2-based scheme is less than that in the hypercube-based scheme.

Proof: According to the analysis on communication overhead addressed in Subsection 5.4 in [3], the result is certainly held.

Figure 5 shows that the comparison on communication overhead between the *H*2-based scheme and the hypercube-based scheme.

5.3. Security Analysis

Here we put focus on two types of attacks against *H*2-based scheme: 1) An adversary may compromise pairwise key between any two nodes or prevent them to establish a pairwise key. 2) The adversary may focus its power to attack against the whole network, for purpose of lowering the probability of pairwise key establishment, or in creasing communication cost.

5.3.1. Attacks against Pairwise Key between Two Nodes

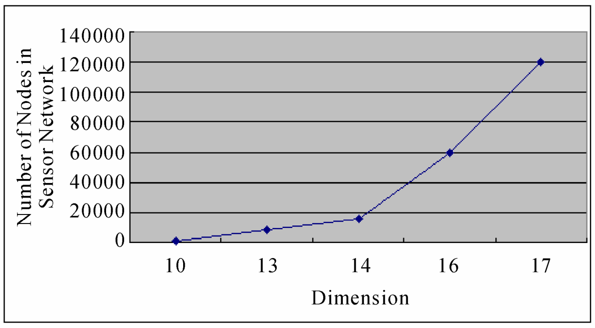

Figure 3. The relationship of dimension *n* and the scale of the sensor networks.

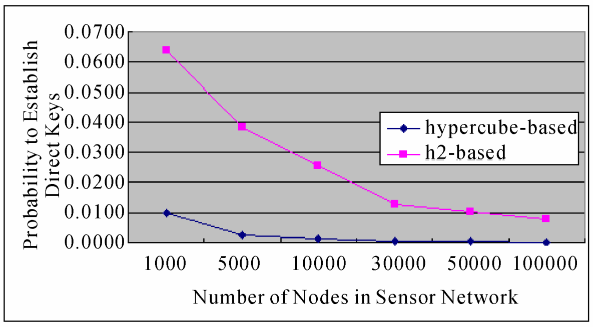

Figure 4. The comparison of probability to establish direct key between H2-based and Hypercube-based algorithms.

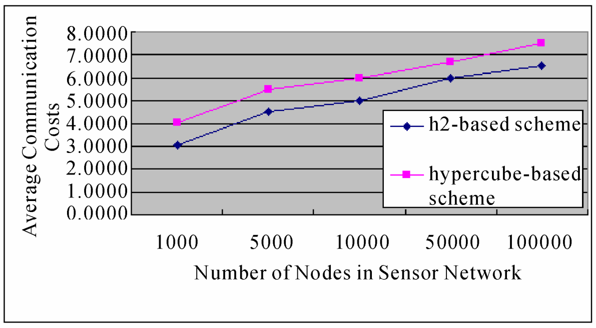

Figure 5. The comparison on average communication overhead between the H2-based and the Hypercube-based schemes.

1) Suppose an adversary launches an attack against two particular nodes, in order to filch their pairwise key. In case that those two nodes are not compromised:

① If the node *u and v* can establish direct pairwise key, the only means to compromise the key is to resolve the polynomial $f(x,y)$, which is shared by the two nodes. As the degree of this polynomial is adversary t, the adversary is required to compromise at least t+1 compromised nodes with the same share of $f(x,y)$.

② If the node u and v need to establish indirect pairwise key, the adversary is required to compromise an intermediate node, or filch the common share of the bivariate polynomial $f(x,y)$ between the two nodes. However, even if the adversary succeeds to achieve the pairwirse key, the nodes u and v can also select alternatives to re-establish key path.

2) Suppose the adversary launch attacks to prevent pairwise key establishment against two particular nodes, denoted as u and v, which are assumed are not compromised. Then the adversary is required to compromise n bivariate polynomials of the node u or v. Notes that to those polynomials are t-degree, which means that if such attacks succeeded, at least $n(t+1)$ nodes should have been compromised.

As addressed above and analysis presented in Subsection 5.5.1 in [3], if an adversary launches attacks against nodes, the security of the *H*2-based scheme if equivalent with that of the hypercube-based scheme. That is, we have the following theorem.

Theorem 13: The security of the *H*2-based scheme if equivalent with that of the hypercube-based scheme.

5.3.2. Attacks against the Whole Network

Suppose that an adversary has known the distribution state of polynomials for each node, he would launch attacks against the whole network systematically by compromising polynomials one by one. Assume that the adversary has compromised l bivariate polynomials, which means that at most $l2^{\lceil n/2 \rceil}$ nodes have been pre-loaded one of those compromised polynomials. However, the rest of the regular nodes, denoted as $N\text{-}l2^{\lceil n/2 \rceil}$ do not contain compromised polynomial shares. That means $N\text{-}l2^{\lceil n/2 \rceil}$ nodes can still work properly. Notice that those regular nodes should avoid to use compromised shares to establish pairwise key.

Clearly, the number of nodes influenced by adversaries in the *H*2-based scheme is more than that in the hypercube-based scheme. However, on the condition that the adversary fails to compromise all of the polynomials, the effected nodes can select other regular nodes to establish pairwise key with others.

In addition, it has proved that the probability of direct key establishment in the *H*2-based scheme is much higher than that in the hypercube-based scheme. Thus in the process of direct key establishment among non-compromised nodes, the degree of the influence cause by adversaries on the *H*2-based scheme is less than that on the hypercube-based scheme. That means the the *H*2-based scheme has ability to secure communications among nodes effectively in sensor networks.

5.3.3. Security Performance

Based on the nice properties of fault tolerance in *H*2-baed scheme, a source node can re-establish pairwise key with the destination by selecting alternative key path.

As addressed in Subsection 4.1, the polynomial pool has $n*2^n$ bivariate t-degree polynomials, that is, $|F|=n*2^n$; As every node contains n different polynomial shares, given a particular share of a bivariate polynomial f, the probability for each node to contain such a share is $n/|F|$. Assume that the number of nodes in a network is $2^{n-1}<N \le 2^n$, and the number of supposed compromised node is N_C, the probability for those compromised nodes to contain i shares of f is

$$P_i = \frac{N_c!}{(N_c-i)!i!}\left(\frac{n}{|F|}\right)^i\left(1-\frac{n}{|F|}\right)^{N_c-i}$$

As the adversary needs to compromise at least t+1 nodes to filch f, the probability of being compromised for f is $P_c=1-\sum_{i=0}^{t} P_i$.

According to Theorem 6, the compromised probability of direct key establishment for any two non-compromised nodes is expressed as $P_{link}= P_c \times P_{H2}$, in case that a particular polynomial f is compromised.

Figure 6 shows the fraction of compromised direct keys between non-comrpomised nodes as a function of the number of compromised keys for H2 and hypercube-based schemes where N=30000 and t=2.

Figure 6 shows that based on the assumption of same network scale and the proportion of compromised nodes, *H*2-based scheme provides higher probability than hypercube-based scheme for direct key establishment between any two non-compromised nodes. *H*2-based scheme would not fail to establish direct key until the proportion increases to 40%, while for *Hypercube*-based scheme, accepted proportion is about 30%.

We further inspect the probability of compromised indirect key. As addressed in Theorem 6, the probability of direct key establishment for any two nodes is $P_{H2} \approx (2^{\lfloor n/2 \rfloor} + 2^{\lceil n/2 \rceil})/(N\text{-}1)$, the probability of indirect key establishment can be expressed as $1\text{-}P_{H.}$ Thus the probability of compromised indirect key is estimated as $(1\text{-}P_{H2})[1\text{-}(1\text{-}\frac{N_C}{N}) \times (1\text{-}P_c)^2]$.

Figure 7 shows that the probability of compromised indirect key between any two non-compromised nodes is a function of the fraction of compromised nodes where N=30000 and t =2.

Figure 7 shows that based on the same conditions of network scale and fraction of compromised nodes, *H*2-based scheme has better performance than hypercube-based scheme on indirect key establishment. The figure also shows that *H*2-based scheme would not fail to establish indirect key until the fraction of compromised nodes rises up to 60%. However, the fraction is only about 40% for *Hypercube*-based scheme.

Here we consider overall security performance of the two schemes. We define the probability of compromised pairwise key (direct or indirect key) is

$$P_{key}=P_{H2}\times P_c+(1-P_{H2})[1-(1-\frac{N_c}{N})\times(1-P_c)^2].$$

Figure 8 shows that the probability of compromised pairwise key is a function of the fraction of compromised nodes where N=30000 and t=2 for the two schemes.

From Figure 8, we can know that the probability of the pairwise key between any two non-compromised nodes when the *H*2-based scheme is applied, is lower than that when the *Hypercube*-based scheme is applied, supposing that the scale and percentage of compromised nodes of the sensor networks are the same.

So, from the above description, it is obvious that the security performance of the *H*2-based scheme is better than that of *Hypercube*-based scheme.

5.4. The Probability of Pairwise Key Re-estab-lishment

A source node has to re-establish key path to the destination once some intermediate nodes have been compromised. According to the previous presented two kinds of dynamic key path establishing algorithms, it is easy to know that the algorithms can find a new alternative key path certainly, when k =1,2 or 3, as long as the distribution of the compromised nodes in the whole sensor network satisfy the conditions of 1,2 or 3- dimensional local-weak-connectivity. Next, lets analyze the probability of pairwise key re- establishment when the distrivution of the compromised node do not satisfy the conditions of 1,2 and 3-dimensional local- weak-connectivities.

According to the pairwise key establishment scheme addressed above, each node in a network is able to communicate $2^{\lfloor n/2\rfloor}+2^{\lceil n/2\rceil}$ nodes to establish direct pairwise key. Assume that the fraction of compromised nodes is p, then the number of non-compromised nodes among $2^{\lfloor n/2\rfloor}+2^{\lceil n/2\rceil}$ is $(1-p)*(2^{\lfloor n/2\rfloor}+2^{\lceil n/2\rceil})$. On the condition that a key path is available among those non-compromised nodes, it's certainly possible for a source node and the destination to establish indirect pairwise key. So, when the distrivution of the compromised node do not satisfy the conditions of 1,2 and 3-dimensional local-weak-

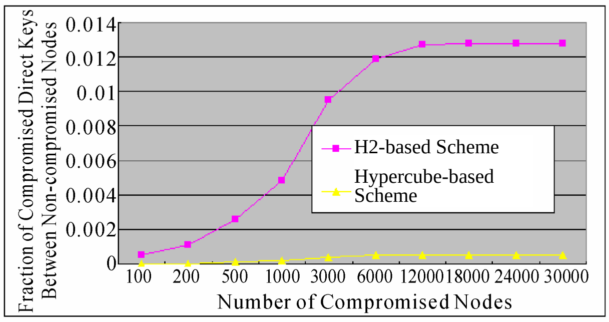

Figure 6. The relation between the fraction of compromised direct keys and the number of compromised nodes in H2- based scheme and Hypercube-based scheme.

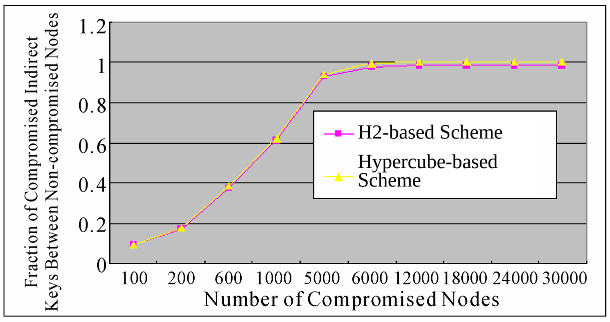

Figure 7. The comparison on the fraction of compromised indirect keys-number of compromised nodes relation between the H2-based scheme and the Hypercube-based scheme.

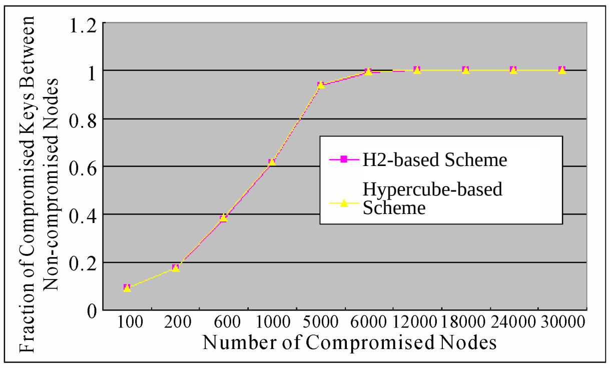

Figure 8. The comparison on the fraction of compromised keys- number of compromised nodes relation between the H2-based scheme and the Hypercube-based scheme.

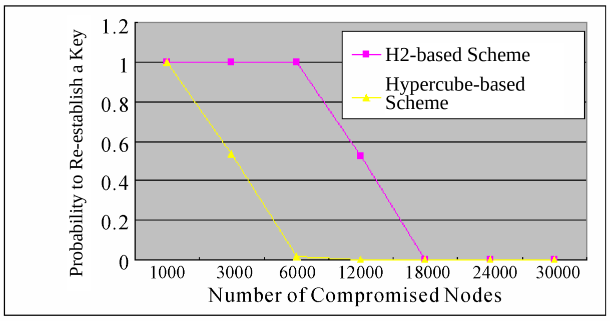

Figure 9. The relation between the probability of re-established keys and the number of compromised nodes in the H2-based and Hypercube-based schemes.

connectivities, the probability of pairwise key re-establishment can be estimated as:

$$P_{re}=1-[1-(1-p)^2(1-P_c)^2]^{(1-p)*(2^{\lfloor n/2\rfloor}+2^{\lceil n/2\rceil})}$$

Assume that N=30000 and t=2, Figure 9 shows that the probability of pairwise key re-establishment is a function of number of compromised nodes in H2 and hypercube-based schemes. It also shows that the probability of pairwise key re-establishment in H2 scheme is higher than that in hypercube-based scheme for any two non-compromised nodes.

6. Conclusions

An H2-based key predistribution scheme is proposed. Compared with polynomial pool-based scheme, it can improve working performance on probability of direct key establishment without additional storage requirement. Moreover, experimental figures show that our algorithm has lower communication cost and more secure than previous related works.

7. Acknowledgment

This work was supported in part by the China Scholarship Council (CSC) in form of a scholarship that funds the author, Lei Wang, studying at Lakehead University as a visiting researcher in Canada.

8. References

[1] L. Eeschnaure and V. D. Gligor, "A key-management scheme for distributed sensor networks," in proceedings of the 9th ACM Conference on Computer and Communication Security, pp. 41–47, 2002.

[2] H. Chan, A. Oerrig, and D. Song, "Random key predistribution schemes for sensor networks," in IEEE Syposium on Research in Security and Privacy, pp. 197–213, 2003.

[3] D. G. Liu, P. Ning, and R. F. Li, "Establishing pairwise keys in distributed sensor networks," ACM Journal Name, Vol. 20, pp. 1–35, 2004.

[4] C. Blundo, A. Desantis, S. Kutten, *et al*., "Perfectly secure key distribution for dynamic conferences," in Advances in Cryptology-CRYPTO'92, LNCS, 740, pp. 471–486, 1993.

[5] L. Wang and Y. P. Lin, "Maximum safety-path matrices based fault-tolerant routing for hypercube multi-computers," Journal of Software, Vol. 15, No. 7, pp. 994–1004, 2004.

[6] L. Wang and Y. P. Lin, "A fault-tolerant routing strategy based on maximum safety-path vectors for hypercube multi-computers," Journal of China Institute of Communications, Vol. 16, No. 4, pp. 130–137, 2004.