Hindawi Publishing Corporation
Journal of Sensors
Volume 2014, Article ID 394147, 14 pages
http://dx.doi.org/10.1155/2014/394147

Research Article

A Portable Random Key Predistribution Scheme for Distributed Sensor Network

Shihui Zheng, Yuan Tian, Lei Jin, and Yu Yang

Information Security Center, National Engineering Laboratory for Disaster Backup and Recovery, Beijing University of Posts and Telecommunications, Beijing 100876, China

Correspondence should be addressed to Shihui Zheng; shihuizh@gmail.com

Received 23 May 2014; Revised 27 August 2014; Accepted 29 August 2014; Published 17 November 2014

Academic Editor: Christos Riziotis

A distributed sensor network (DSN) can be deployed to collect information for military or civilian applications. However, due to the characteristics of DSNs such as limited power, key distribution for a distributed sensor network is complex. In this paper, a neighbor-based path key establishing method and a seed-based algorithm are put forward to improve the original random key pre-distribution scheme. The new scheme is portable because it is independent of the routing protocol. Moreover, the connectivity of the entire network also approaches 1. In particular, the new scheme can keep high connectivity by setting a small amount of redundancy in parameter values when the number of neighbors drops because of the node dormancy or death. The resilience against node capture in our scheme is not lower than that in the l-path scheme and the basic schemes when the number of hops in a path is larger than 5, and the simulation result shows that the efficiency of our scheme is also slightly higher.

1. Introduction

A lot of secure key distribution techniques, for instance, the public cryptography, cannot be applied in Wireless sensor networks (WSNs) due to the characteristics of WSNs like large scale network, lacking of trusted infrastructure, and physical constraints to energy and memory. A naive solution is that a single master key is used in all communications which is too vulnerable under the node capture attack. Matsumoto and Imai [1] defined key predistribution (KPD) which is the method of distribution of keys onto nodes before deployment. Therefore, the nodes build up the network using their secret keys after deployment, that is, when they reach their target position. Blom [2] (see also [3]) presented λ-secure key predistribution schemes. The tradeoff is that, unlike the $(N-1)$ pairwise-key scheme, those schemes are no longer perfect against node capture. Instead, the λ-secure KPD schemes [2–7] have perfect security when the number of captured nodes is less than λ. However, once more than λ nodes are controlled by a specific attacker, all keys in the entire network are compromised. Deployment knowledge based KPD schemes [8–10] utilized deployment knowledge to improve the connectivity and the resilience against node capture. However, these solutions are not quite viable since the location of each node may be unknown before deployment. Local center based scheme [11–13] is another type of general KPD techniques, assuming that the network consists of a local trusted infrastructure. Some trusted and powerful nodes distribute keys to sensor nodes around them. However, it is not accessible in a random deployment and dynamic network. In WSNs, random key predistribution (RKPD) schemes [14–21] are widely accepted due to its simplicity, low overhead, scalability, and high global connectivity.

1.1. Related Work. In 2002, Eschenauer and Gligor [14] proposed a random key predistribution (RKPD) scheme, often called basic scheme, using giant component theory. The basic scheme includes 3 phases: (1) key predistribution, (2) shared key discovery, and (3) path key establishment. The first phase is executed before the deployment of nodes. A controller generates a large key pool, and each node draws keys (to form a key ring) out of the pool without replacement. The second phase may occur at the neighbor discovery phase. Two neighbor nodes (two nodes connected physically) try to find a common key identifier in their key rings. If such a key identifier is found, these nodes are logically connected and the key corresponding to the same key identifier is called a shared key. Then, the two nodes separately add a record including the other party's identifier and the shared keys' identifier to a list, called a neighbor list. The last phase occurs just before the information transmission. A pair of nodes x and y (including nodes which are only physically connected with x) try to find a path $(x, w_1), (w_1, w_2), \ldots, (w_{t-1}, w_n), (w_t, y)$, where (i, j) represents that sensor nodes i and j are logically connected neighbors. Afterwards, x and y can distribute a key on the path, and the key is called a path key. The basic scheme can be parameterized to meet the demands that the connected probability of the entire network closes to 1.

Later, Chan et al. [15] gave a q-composite scheme in which any two logically connected nodes need to share q keys instead of only one to establish a path key. An l-path scheme was also presented in [15], in which the path key is broken up into l nuggets, and these nuggets are passed to y along l disjoint paths. But the problem of discovering multiple disjoint paths is computationally hard, and too much overhead may be incurred in this process. Zhu et al. [18] relaxed the requirement where a single physical path is used as long as the nuggets of the path key are transmitted through multiple logically disjoint paths. However, the cost of discovering multiple logically disjoint paths is also expensive.

Gu et al. [20] pointed that the performance of the basic scheme is satisfactory only in highly dense sensor networks, where the average number of physical neighbors per node is more than 20. They proposed a methodology called network decoupling and designed a new protocol to establish the path key between physical neighbors with the help of a node proxy. Li et al. [21] also proposed a multihop proxies random key predistribution scheme. In their schemes, each node constructs a local logical graph and a local physical graph before the path-key establishment, which incurs a large amount of storage, computation, and communication overload. In addition, the physical graph of WSNs changes all the time when the node moves or dies. Thus, if the local graph is not updated in time, the path key establishment will fail.

1.2. Our Contributions. Generally, a key establishment scheme has nothing to do with the routing protocol. However, the path in lots of RKPD schemes, such as the basic scheme, is set up through a routing protocol. That is, a customized routing protocol needs to be used along with those key distribution schemes, which will seriously affect the portability of those RKPD schemes. Moreover, the number of hops of the path may increase because the adjacent nodes in the path must be logically connected (see Section 6). In fact, the total amount of computation cost for deciding whether a shared key exists in key rings of two adjacent nodes is also very huge because the routing process may involve many nodes (see Section 6).

A new method to establish the path key with the help of neighbors is presented in this paper. The basic steps of Phase 3 are changed, and a path that the adjacent nodes only need to be physically connected can be achieved by original routing protocols [22]. To reduce the communication overload, a seed-based method is used to chose key ring for each node upon the input of node's identifier. Although the seed-based method has been mentioned in paper [18, 19], we firstly construct a deterministic algorithm **G**, with which the times that each key is selected by nodes approximate the average value. In addition, the connectivity of the new scheme is also higher if suitable parameters are chosen. Specifically, when the number of neighbors is less than the predefined out-degree due to the node dormancy or death, the probability that the entire network is connected still approaches 1 by setting a small amount of redundancy in parameters. The probability that a link is compromised when nodes are captured in our scheme is equal to that of the basic scheme, and it is higher than that of the q-composite scheme and that of the l-path scheme. However, the probability that a path key is compromised in the establishment phase is lower than that of l-path schemes when the number of hops is more than 4. Our scheme is less competitive in terms of the computation and communication complexity analysis. However, the execution time of a path key establishment for our scheme is slightly less than that of the basic scheme in the simulation. The reason is that, the complexity does not include the extra traffic and calculations that are caused by the customized routing protocol in the basic scheme.

2. The New RKPD Scheme

In the rest of this paper, N represents the number of nodes in the entire wireless sensor network, and each node has a node identifier i, $(i = 1, 2, \ldots, N)$. L is the number of keys in the key pool, and each key also has a key identifier k, $(k = 1, 2, \ldots, L)$. n is the average number of nodes in single-hop communication range, called node density. And s is the number of keys in the key ring for each node. In addition, t expresses the number of hops of a path between the source node x and the destination node y.

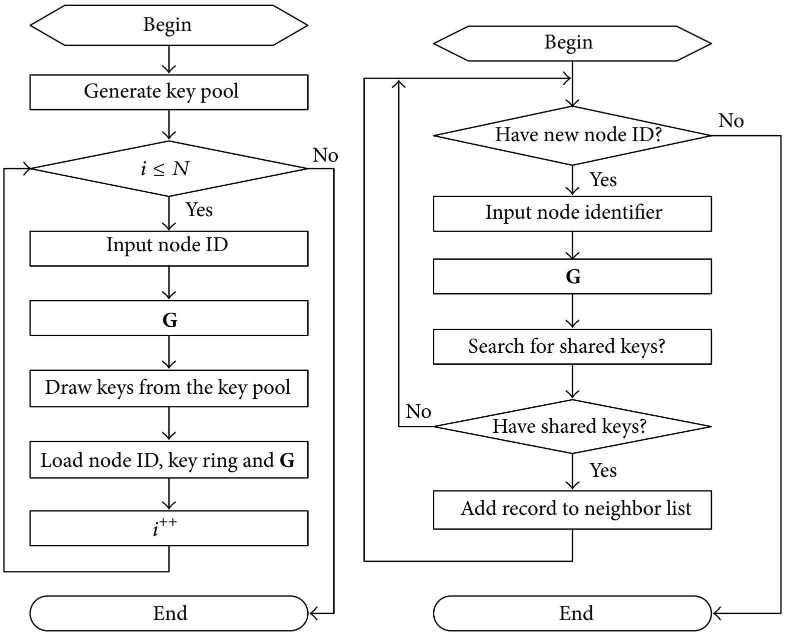
2.1. The Scheme. A new random key predistribution scheme is described in this section. The scheme also includes three phases: (1) key predistribution, (2) shared key discovery, and (3) path key establishment.

Phase 1 (key predistribution). Before the deployment of nodes, for each node, a control center (CC) randomly chooses a key ring and loads it into the node (the flow chart is shown in Figure 1).

Step 1. The CC randomly generates L keys and assigns a unique key identifier k $(k \in [1, L])$ to each key_k; those keys and the corresponding identifiers compose a key pool.

Step 2. The CC chooses a deterministic algorithm **G** to decide the key identifiers allocated to each node on the input of the node's identifier.

Step 3. For each node i $(i = 1, \ldots, N)$, the CC inputs its identifier i into **G** and output s distinct values between 1 and L, denoted by $\{T_1, T_2, \ldots, T_s\}$. At last, the CC draws s keys whose key identifiers are $\{T_l \mid l = 1, \ldots, s\}$. Those keys $\{\text{key}_{T_l} \mid l = 1, \ldots, s\}$ and the corresponding key identifiers



(a) Key predistribution

(b) Shared key discovery

Figure 1: The operation process of Phases 1 and 2.

$\{T_l \mid l = 1, \ldots, s\}$ compose a key ring which is loaded into node i. Also the algorithm **G** is loaded into each node.

Phase 2 (shared key discovery). After the deployment of nodes, each node creates its own neighbor list.

Step 1. Each node i $(i = 1, \ldots, N)$ broadcasts its identifier and records the received identifiers, denoted by $\{i_r \mid r = 1, 2, \ldots\}$.

Step 2. For each node i_r, node i runs the procedure **G** and generates the key identifier set of nodes i_r. If there is a common key identifier in such set and its own key ring, they are logically connected. Then node i adds a record involving the node identifier i_r and the same key identifier to its neighbor list.

Phase 3 (path key establishment). Node x wants to establish a path key with node y. If they are in wireless communication range and have a shared key, that is, they are on each other's neighbor list, the shared key can be used as the path key. Else, x randomly generates a path key pk and encrypts pk with some key in y's key ring. The ciphertext, denoted by $\mathrm{CT} = E(pk)$, together with the identifier of the encryption key is sent to y on a physical connected path founded by a routing protocol [22]. Finally, y finds the encryption key corresponding to the received key identifier and decrypts CT to obtain pk. The following procedure explains how x can find an encryption key which belongs to y's key ring. Figure 2 shows the process of source node x and Figure 3 shows the process of destination node y and x's neighbors z.

Step 1. x inputs y's node identifier i_y into the procedure **G** to generate y's key identifier set and then searches for a shared key identifier between y's key identifier set and its own's in its key ring using binary search method. If a same key identifier k_{xy} is found, x looks up k_{xy} in its own key ring and obtains the encryption key. Else, if all the key identifiers are different, x goes to step 2.

Step 2. x broadcasts y's identifier i_y.

Step 3. x's neighbor z_j which receives the identifier looks i_y up in its neighbor list. If x's identifier i_x is not on the list, z_j stops.

Step 4. z_j inputs y's identifier i_y into the procedure **G** to generate y's key identifiers and then tries to find a shared key identifier between y's key identifiers and its own's. If z_j does not find a same key identifier, it stops. Otherwise, z_j sends "1" to x for confirmation.

Step 5. x chooses z^* from neighbors which have responded and obtains the shard key$_{k_{z^* x}}$ through a neighbor list query and a key ring query. Then, x encrypts pk with the shared key and sends the temp ciphertext $\mathrm{CT}_{\mathrm{temp}}$ and the key identifier $k_{z^* x}$ to z^*.

Step 6. z^* searches its key ring for the shared key$_{k_{z^* x}}$ and decrypts $\mathrm{CT}_{\mathrm{temp}}$ to obtain pk. After that, z^* searches its key ring for the shared key$_{k_{z^* y}}$ and encrypts pk with key$_{k_{z^* y}}$. At last, z^* sends the final ciphertext CT and the encryption key identifier $k_{z^* y}$ to x.

Step 7. x forwards CT and $k_{z^* y}$ to y.

2.2. The Algorithm **G**. The deterministic algorithm **G** is used to decide the key ring allocated to each node. Specifically, for

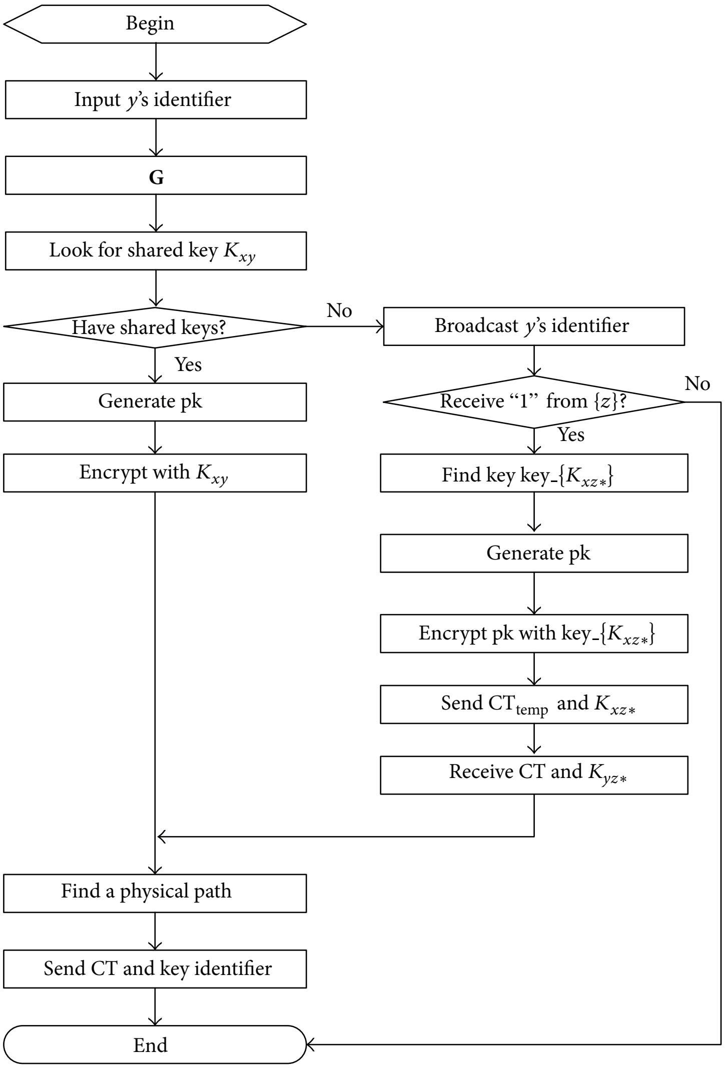

Begin
Input y's identifier
G
Look for shared key K_{xy}
Have shared keys?
No
Yes
Broadcast y's identifier
Receive "1" from $\{z\}$?
No
Yes
Generate pk
Encrypt with K_{xy}
Find key key_$\{K_{xz*}\}$
Generate pk
Encrypt pk with key_$\{K_{xz*}\}$
Send $\mathrm{CT}_{\mathrm{temp}}$ and K_{xz*}
Receive CT and K_{yz*}
Find a physical path
Send CT and key identifier
End

Figure 2: The operation process of node x in Phase 3.

each node with a unique node identifier, the algorithm generates s distinct integers between 1 and L using pseudorandom number generator upon the input of a node identifier. These s integers are the identifiers of the keys for this node. In this paper, the algorithm **G** is based on a bit-oriented linear feedback shift registers (LFSR). The content of the LFSR is denoted by $u_0, u_1, \ldots, u_{\alpha-1}$. The feedback polynomial of the LFSR, $f(x)$ is a primitive polynomial of degree α.

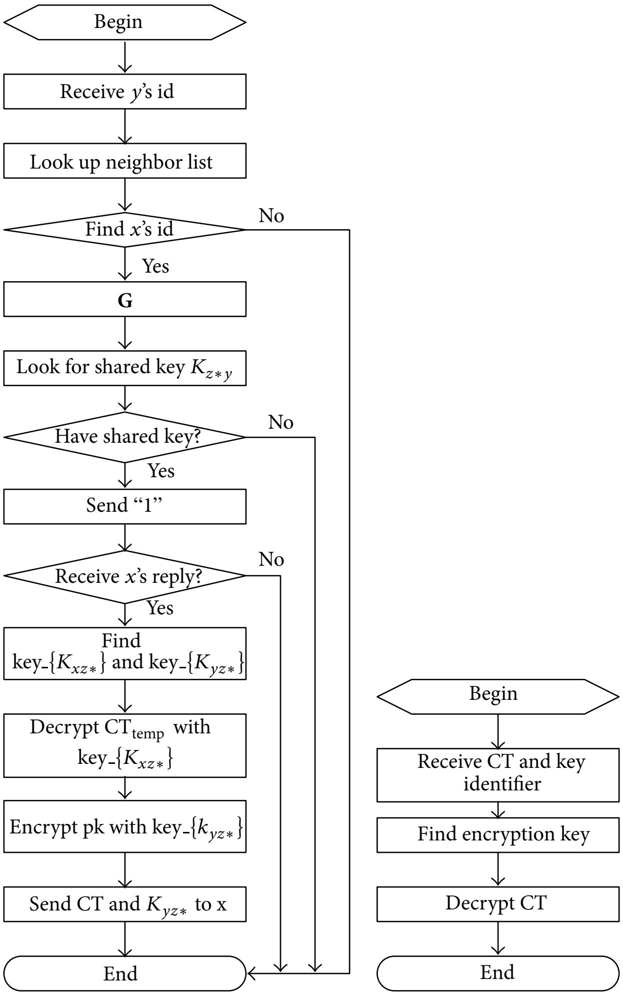
2.2.1. Initialization. Before any bit stream is generated, the register must be initialized with the node identifier. Let the bits of the node identifier k be denoted by k_j, $0 \le j < \lceil \log_2 N \rceil$. The initialization phase is done as follows. First, load the LFSR with the node identifier bits, $u_j = k_j$, $0 \le j < \lceil \log_2 N \rceil$, and then the remaining bits of the LFSR are filled with "1"s.

2.2.2. Key Identifier Set Generation. Create a one-dimensional array T of length s and initialize the array with "0"s. After the following steps, output the array which is fulfilled with key identifiers for a node.

```
For each $j \in [0, s-1]$

    (1) the LFSR shift to producing bit stream;
    (2) once every $\lceil \log_2 L \rceil$ bits $v_\beta, v_{\beta+1}, v_{\beta+2}, \ldots, v_{\beta+\lceil \log_2 L \rceil}$ have been generated,
        compute value $v = 2^{\lceil \log_2 L \rceil - 1} \cdot v_{\beta+\lceil \log_2 L \rceil} + \cdots + 2 \cdot v_{\beta+1} + v_\beta$;
    (3) if $v > L$ or $v = T[\gamma]$ $(\gamma = 0, \ldots, j)$, go to (1);
    (4) else, $T[j] = v$;
    (5) sort the sequence $T[0], T[1], \ldots, T[j]$ using standard insertion sort method

end for.
```


Begin
Receive y's id
Look up neighbor list
Find x's id
No
Yes
G
Look for shared key K_{z*y}
Have shared key?
No
Yes
Send "1"
Receive x's reply?
No
Yes
Find key_{K_{xz*}} and key_{K_{yz*}}
Decrypt $\mathrm{CT}_{\mathrm{temp}}$ with key_{K_{xz*}}
Encrypt pk with key_{k_{yz*}}
Send CT and K_{yz*} to x
End
Begin
Receive CT and key identifier
Find encryption key
Decrypt CT
End

(a) The process of neighbor z

(b) The process of y

FIGURE 3: The operation process of nodes z and y in Phase 3.

2.2.3. Design Criteria

(1) The Number of Nodes, In Which Each Key in the Key Pool is Allocated to, Closes to sN/L. In random sampling without replacement, each member of a population has an equal chance of being included in the sample. Hence, each key in the key pool has a probability of s/L to be chosen by a node. Since the network size is N, each key should be statistically allocated to sN/L nodes. In practice, however, using computational algorithms can only produce long sequences which seems to be random because the outputs are in fact determined by a shorter initial value, called a seed. So the generators are regarded as pseudorandom number generators. As a result, the number of nodes that each key is selected by can only close to sN/L in reality.

A binary sequence generated by a linear feedback shift which registers with a primitive feedback polynomial has balance property, run property, and correlation property. Therefore, any truncation of the sequence can be regarded as a pseudorandom number. The experimental result (see Figure 5) also shows that the times in which each key identifier is selected approximate to sN/L.

(2) $\alpha \geq \max(\lceil \log_2 N \rceil, \lceil \log_2 L \rceil)$. First of all, $\alpha > \lceil \log_2 N \rceil$ ensures that the register can be initialized to distinct states according to distinct node identifiers. Moreover, if $\alpha \leq \lceil \log_2 L \rceil$, the key identifiers (in binary format) that have a run of length e ($e > \alpha$) will never be selected by **G** (refer to run test of LFSR for the proof [23]).

(3) $\alpha < 3 \cdot \lceil \log_2 L \rceil$. This condition guarantees that a long subsequence of "0" will not appear at the start of the sequence, since most of those bits will be discarded in step 4 of the key identifier set generation procedure.

(4) $2^{\alpha} - 1 \gg \lceil \log_2 L \rceil \cdot \lceil \log_2 N \rceil$. This condition ensures that the key identifier set for a node can be generated in one cycle of the sequence produced by the LFSR.

Example 1. Suppose $N = 1000$, $L = 100$, and $s = 12$. According to the design criteria, we choose $\alpha = 12$ and the

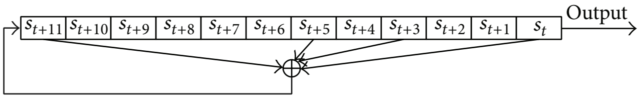


Figure 4: The structure of the LFSR.

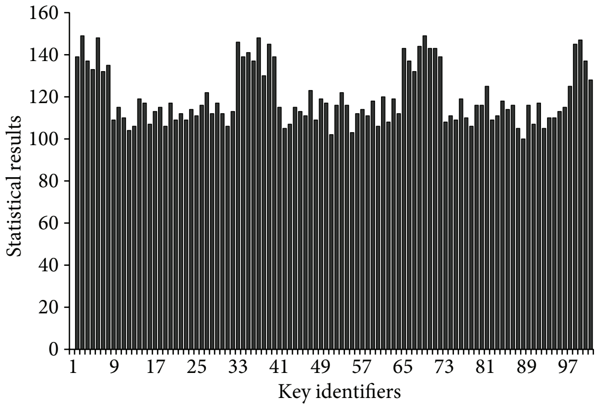


Figure 5: The times of each key identifier being chosen.

feedback polynomial of the LFSR $f(x) = x^{12} + x^6 + x^4 + x + 1$ (see also Figure 4). Obviously, α meets the conditions (2), (3), and (4).

The average times that each key is drawn by nodes are $sN/L = 12000/100 = 120$, and the actual statistical results are shown in Figure 5. It can be seen that most values are between 100 and 140, which close to 120.

3. Connectivity

The connected probability p refers to the probability that any two nodes in a distributed sensor network can establish a path key successfully. Next, we will prove that the connected probability p of the new RKPD scheme is

$$p = 1 - \frac{C_L^s \cdot C_{L-s}^s}{\left(C_L^s\right)^2} \cdot \left(1 - \frac{\sum_{i=1}^{s-1} \sum_{j=1}^{s-i} C_s^i \cdot C_s^j \cdot C_{L-2s}^{s-i-j}}{C_L^s}\right)^{n-1}. \tag{1}$$

As mentioned in Phase 3, a path key can be securely sent to y as long as an encryption key can be found, and the probability of which will be discussed in two situations.

Case 1. x finds a shared key k_{xy} with y in step 1 of Phase 3, and this key is used as the encryption key.

Each key ring of size s is randomly drawn out of the key pool, so the probability that two nodes do not share any key is $\overline{P_{xy}} = C_L^s \cdot C_{L-s}^s/(C_L^s)^2$. In other words, the first key ring is picked at random and the second key ring is drawn out of the remaining $L-s$ unused keys in the pool. Therefore, at least one shared key exists with probability $P_{xy} = 1-(C_L^s \cdot C_{L-s}^s/(C_L^s)^2)$.

Case 2. x cannot find any shared key with y in step 1 of Phase 3, but a logically connected neighbor z^* which has a shared key k_{yz^*} with y can be found. Hence, that shared key will be used as the encryption key.

Table 1: System parameters in the new scheme when $p = 0.9999$.

n/L	100	1000	10000	100000
10	12	38	119	380
20	10	30	97	310
30	9	27	86	270
40	8	25	77	250
50	8	23	72	230
60	7	22	69	220

A neighbor of node x, denoted by z, has both a shared key with x and a shared key with y which means that i $(1 \le i \le s-1)$ keys of z's key ring are taken from x's ring and j $(1 \le j \le s-i)$ keys are taken from y's ring. So the total number of the possible key rings of z is $\sum_{i=1}^{s-1} \sum_{j=1}^{s-i} C_s^i \cdot C_s^j \cdot C_{L-2s}^{s-i-j}$. Therefore, such a neighbor exists with probability $P_{xyz} = \sum_{i=1}^{s-1} \sum_{j=1}^{s-i} C_s^i \cdot C_s^j \cdot C_{L-2s}^{s-i-j}/C_L^s$.

In addition, the node density is n, so there are $n-1$ neighbors in average. Therefore, the probability that at least one neighbor can help x encrypt pk in step 5 and step 6 is $P_{xyz,n} = 1-(1-P_{xyz})^{n-1}$. In sum, the connected probability is

$$\begin{aligned} p &= \Pr[\text{Case }1] + \Pr\left[\text{Case }2 \mid \overline{\text{Case }1}\right] \\ &= P_{xy} + \overline{P_{xy}} \cdot P_{xyz,n} = \left(1 - \frac{C_L^s \cdot C_{L-s}^s}{\left(C_L^s\right)^2}\right) \\ &\quad + \frac{C_L^s \cdot C_{L-s}^s}{\left(C_L^s\right)^2} \cdot \left[1 - \left(1 - \frac{\sum_{i=1}^{s-1} \sum_{j=1}^{s-i} C_s^i \cdot C_s^j \cdot C_{L-2s}^{s-i-j}}{C_L^s}\right)^{n-1}\right] \\ &= 1 - \frac{C_L^s \cdot C_{L-s}^s}{\left(C_L^s\right)^2} \cdot \left(1 - \frac{\sum_{i=1}^{s-1} \sum_{j=1}^{s-i} C_s^i \cdot C_s^j \cdot C_{L-2s}^{s-i-j}}{C_L^s}\right)^{n-1}. \end{aligned} \tag{2}$$

According to (2), the system parameters can be decided. For instance, let $p = 0.9999$ while the parameters are listed in Table 1. The first row is the size of key pool and the first column is the node density. Each entry of the table is the theoretical values of the key ring size.

Let the total number of nodes in the network be $N = 1000$ and let the connected probability be $p = 0.9999$. Figure 6 shows how the key ring size s varies along with the node density n.

In the figures of this section, symbol (O) denotes the new scheme, symbol (B) denotes the basic scheme, and symbol (Q) denotes the q-composite scheme. It can be seen that the theoretical value of s in the basic scheme and the q-composite scheme increases sharply when n closes to 20. The reason is that the predefined out-degree is $d = 20$. In other words, if the

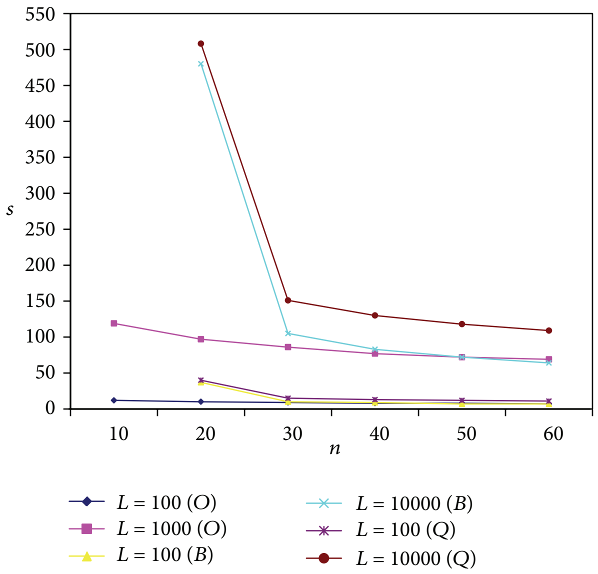


Figure 6: Changes of the key ring size.

node density n is equal to or less than d, the connectivity will be seriously affected. Since the out-degree cannot be modified when nodes are deployed, the size of the key ring must be very large to keep high connectivity when nodes sleep or die. As a result, the amount of storage per node in the basic scheme and the q-composite scheme will increase. However, the ring size in the new scheme increases smoothly as the node density decreases. Thus, by adding a small amount of redundancy to the key ring size, the network will work well even in the case in which the number of active nodes is less than d. For example, let $s = 12$ when $L = 100$.

4. Security

Resilience in WSNs refers to the resistance of key distribution schemes against node capture. When sensor nodes are deployed in hostile areas (e.g., battle surveillance), an adversary can mount a physical attack on a sensor node and recover secret information from its memory. So we are interested in the question: for any two nodes x and y which have not been captured by the attacker, what is the probability that the attacker can eavesdrop their communications using the subset of the key pools that was recovered from the nodes captured. That is, the shared key of two physically connected neighbors (also called a link) is in the compromised key set or a path key is compromised in path key establishment.

Theorem 2. *The probability that any secure link in the shared-key discovery phase between two uncompromised nodes is compromised when m nodes have been captured is* $\sum_{\tau=s}^{\min(ms,L)}[C_L^\tau \cdot (\tau/L) \cdot (((C_\tau^s)^m - \sum_{\lambda=1}^{\tau-s}(-1)^{\lambda+1}C_\tau^\lambda (C_{\tau-\lambda}^s)^m)/(C_L^s)^m)]$.

Proof. Let $I \subset \{1, 2, \dots, L\}$ be a subset of the key identifers of the key pool, where $s \le |I| \le \min(ms, L)$. And let C_I be the event in which key with identifier $k_j \in I$ is compromised after m nodes are captured by an adversary. The probability $\Pr[C_I]$ is $((C_{|I|}^s)^m - \sum_{\lambda=1}^{|I|-s}(-1)^{\lambda+1}C_{|I|}^\lambda (C_{|I|-\lambda}^s)^m)/(C_L^s)^m$ (please refer to paper [24] for the proof).

Let C be the event in which the link between the two nodes is compromised, so $\Pr[C] = \sum_I \Pr[C \mid C_I] \cdot \Pr[C_I]$. If I is fixed, a link in our scheme is compromised if and only if the shared key's identifier of the two nodes is in I. Hence, $\Pr[C \mid C_I] = |I|/L$.

From the above two aspects, the probability of compromising a link in the shared-key discovery is

$$\begin{aligned}\Pr[C] &= \sum_I \Pr[C \mid C_I] \cdot \Pr[C_I] \\ &= \sum_I \left[\frac{|I|}{L} \cdot \left(\frac{\left(C_{|I|}^s\right)^m - \sum_{\lambda=1}^{|I|-s}(-1)^{\lambda+1}C_{|I|}^\lambda \left(C_{|I|-\lambda}^s\right)^m}{\left(C_L^s\right)^m}\right)\right] \\ &= \sum_{\tau=s}^{\min(ms,L)}\left[C_L^\tau \cdot \frac{\tau}{L} \right. \\ &\qquad \left. \cdot \left(\frac{(C_\tau^s)^m - \sum_{\lambda=1}^{\tau-s}(-1)^{\lambda+1}C_\tau^\lambda (C_{\tau-\lambda}^s)^m}{(C_L^s)^m}\right)\right].\end{aligned} \tag{3}$$

The probability that a link is compromised in our scheme is equal to that in the basic scheme, and it is higher than that of the q-composite scheme. □

Theorem 3. *The probability that a path key between two nodes is compromised in the path key establishment phase when m nodes have been captured is* $\sum_{\tau=s}^{\min(ms,L)}[C_L^\tau \cdot (P_{xy} \cdot (\tau/L) + (1 - P_{xy}) \cdot (2 \cdot (\tau/L) - (\tau/L)^2)) \cdot (((C_\tau^s)^m - \sum_{\lambda=1}^{\tau-s}(-1)^{\lambda+1}C_\tau^\lambda (C_{\tau-\lambda}^s)^m)/(C_L^s)^m)]$.

Proof. In path key establishment, let B be the event in which a path key between the two nodes is compromised in the path key establishment phase when m nodes have been captured. Similarly, $\Pr[B] = \sum_I \Pr[B \mid C_I] \cdot \Pr[C_I]$.

In our scheme, a path key is encrypted with one shared $\text{key}_{k_{xy}}$, named direct encryption, with probability P_{xy} (see Figure 7) and is encrypted with two shared keys $(\text{key}_{k_{xz^*}}, \text{key}_{k_{yz^*}})$, named indirect encryption, with probability $1 - P_{xy}$ (see Figure 8). We already know that a shared key is compromised if and only if the shared key's identifier is in I. So in the direct encryption, the path key is compromised with probability $|I|/L$. And in the indirect encryption, the probability that the path key is compromised is the probability of compromised one shared key plus the probability of compromised another shared key minus the probability of compromising both shared keys. □

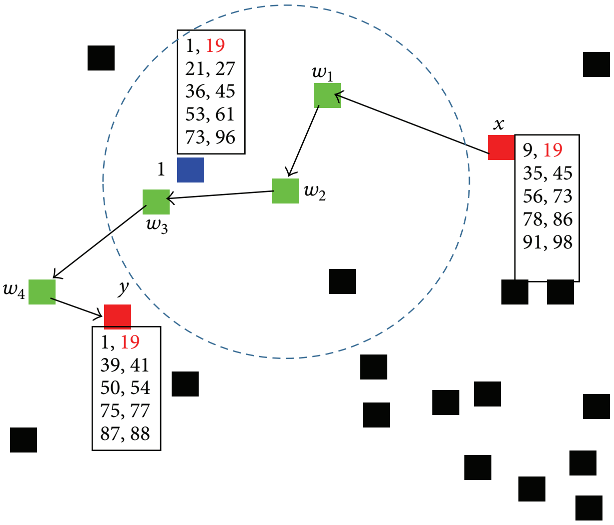


Figure 7: The path key is compromised in the direct encryption. ^1x is source node and y is destination node. The attacker captures a node (the blue node) whose key ring includes key identifier 19. So the path key between x and y may be eavesdropped at anywhere near the path (x-w_1-w_2-w_3-w_4-y), for example, at the link (w_2-w_3).

So the probability that the path key is compromised is $\Pr[B \mid C_I] = P_{xy}\cdot(|I|/L)+(1-P_{xy})\cdot(2\cdot(|I|/L)-(|I|/L)^2)$ when I is fixed. To sum up, in our scheme, the probability that the path key is compromised is

$$\begin{aligned}
\Pr[B] &= \sum_{I} \Pr[B \mid C_I]\cdot\Pr[C_I] \\
&= \sum_{I}\Bigg[\left(P_{xy}\cdot\frac{|I|}{L}+\left(1-P_{xy}\right)\cdot\left(2\cdot\frac{|I|}{L}-\left(\frac{|I|}{L}\right)^2\right)\right) \\
&\quad\cdot\left(\frac{\left(C_{|I|}^{s}\right)^m-\sum_{\lambda=1}^{|I|-s}(-1)^{\lambda+1}C_{|I|}^{\lambda}\left(C_{|I|-\lambda}^{s}\right)^m}{\left(C_L^s\right)^m}\right)\Bigg] \\
&= \sum_{\tau=s}^{\min(ms,L)}\Bigg[C_L^{\tau}\cdot\left(P_{xy}\cdot\frac{\tau}{L}+\left(1-P_{xy}\right)\right. \\
&\quad\left.\cdot\left(2\cdot\frac{\tau}{L}-\left(\frac{\tau}{L}\right)^2\right)\right) \\
&\quad\cdot\left(\frac{\left(C_{\tau}^{s}\right)^m-\sum_{\lambda=1}^{\tau-s}(-1)^{\lambda+1}C_{\tau}^{\lambda}\left(C_{\tau-\lambda}^{s}\right)^m}{\left(C_L^s\right)^m}\right)\Bigg].
\end{aligned} \tag{4}$$

In the basic scheme [14] and the q-composite [15] scheme, the path key is established on a multihop path (mainly refers to multiple links using different encryption key), so the path key is compromised when at least one hop is compromised (see Figure 9). Assuming that one hop (a link) is compromised with probability $\Pr[C \mid C_I]$ when m nodes are captured, the probability that all t-hops are secure is $(1-\Pr[C \mid C_I])^t$. As a result, the probability that the path key is compromised is $\Pr[B] = \sum_{\tau=s}^{\min(ms,L)}[C_L^{\tau}\cdot(1-(1-\Pr[C \mid C_I])^t)\cdot(((C_{\tau}^{s})^m-\sum_{\lambda=1}^{\tau-s}(-1)^{\lambda+1}C_{\tau}^{\lambda}(C_{\tau-\lambda}^{s})^m)/(C_L^s)^m)]$.

As a result, in the basic scheme, the probability that the path key is compromised is $\Pr[B] = \sum_{\tau=s}^{\min(ms,L)}[C_L^{\tau}\cdot(1-(1-(\tau/L))^t)\cdot(((C_{\tau}^{s})^m-\sum_{\lambda=1}^{\tau-s}(-1)^{\lambda+1}C_{\tau}^{\lambda}(C_{\tau-\lambda}^{s})^m)/(C_L^s)^m)]$. And in the q-composite scheme, the probability that the path key is compromised is $\Pr[B] = \sum_{\tau=s}^{\min(ms,L)}[C_L^{\tau}\cdot(1-(1-\sum_{j=q}^{s}(C_{\tau}^{j}/C_L^{j})\cdot(p(j)/p'))^t)\cdot(((C_{\tau}^{s})^m-\sum_{\lambda=1}^{\tau-s}(-1)^{\lambda+1}C_{\tau}^{\lambda}(C_{\tau-\lambda}^{s})^m)/(C_L^s)^m)]$, where $p(i) = (C_L^i)\cdot(C_{L-i}^{2(s-i)})\cdot(C_{2(s-i)}^{s-i})/(C_L^s)^2$ and $p' = d/n$. Please refer to paper [24] for the proof of $\Pr[C \mid C_I]$ in the basic scheme and the q-composite scheme.

In the l-path scheme [15], a path key is compromised if and only if l disjointed path is all compromised, that is, $\Pr[C \mid C_I] = \prod_{i=1}^{l}(1-(1-\Pr[C \mid C_I])^{t_i})$, where t_i is the number of hops on the ith path. So the probability that the path key is compromised is $\Pr[B] = \sum_{\tau=s}^{\min(ms,L)}[C_L^{\tau}\cdot\prod_{i=1}^{l}(1-(1-(\tau/L))^{t_i})\cdot(((C_{\tau}^{s})^m-\sum_{\lambda=1}^{\tau-s}(-1)^{\lambda+1}C_{\tau}^{\lambda}(C_{\tau-\lambda}^{s})^m)/(C_L^s)^m)]$.

The probability that a path key will be compromised in the establishment phase when $m = 1$ is listed in Table 2 ($L = 100$, $s = 10$, and $q = 2$). It can be seen that the probability in our scheme will be lower than that of the l-path scheme when the number of hops is larger than 4.

5. Complexity

In this section, the storage, computation, and communication complexity of the new scheme are presented.

(1) Storage Complexity. Same as the basic scheme, each node in the new scheme stores a key ring and a neighbor list. There are s key-identifier/key pairs in the key ring. Let |key| represent the length of a key, so the key ring takes $(s\cdot|\text{key}|+s\cdot\lceil\log_2 L\rceil)$ bits of storage space. On the other hand, the probability that two neighbor nodes are logically connected is also P_{xy}, and the average number of neighbors for a node is $n-1$. Thus, there are about $[P_{xy}\cdot(n-1)]$ records on the neighbor list. Each record includes a node identifier and a key identifier. Therefore, the neighbor list uses $[P_{xy}\cdot(n-1)\cdot(\lceil\log_2 N\rceil+\lceil\log_2 L\rceil)]$ bits memory space. Consequently, the total storage complexity of the new scheme is $O(s\cdot|\text{key}|+s\cdot\lceil\log_2 L\rceil+P_{xy}\cdot(n-1)\cdot(\lceil\log_2 N\rceil+\lceil\log_2 L\rceil))$.

(2) Communication Complexity. Let |CT| be the length of a ciphertext. If x and y share a key, the main message transmitted on the channel is the ciphertext and the key identifier of the encryption key. So, the communication complexity is $O((t+1)\cdot(\lceil\log_2 L\rceil+|\text{CT}|))$, where t represents the number of hops in a path.

If x and y do not share any keys, the process of searching for an encryption key (step 2 to step 6) will bring extra traffic. y's identifier with length $\lceil\log_2 N\rceil$ is sent in step 2, and a key identifier and a ciphertext are transmitted both in step 5 and in step 6. The probability that a neighbor z has shared keys both with x and with y is $P_{xy}\cdot P_{xy}$, so about $[(n-1)\cdot P_{xy}\cdot P_{xy}]$ bits confirmation message will be sent to x

TABLE 2: The probability that a path key is compromised.

t	1	2	3	4	5	6
Our scheme	0.129743	0.129743	0.129743	0.129743	0.129743	0.129743
Basic scheme	0.100000	0.190000	0.271000	0.343900	0.409510	0.468559
q-composite scheme	0.003741	0.007469	0.011183	0.014882	0.018568	0.022240
l-path scheme	0.010000	0.036100	0.073441	0.0118267	0.167698	0.219548

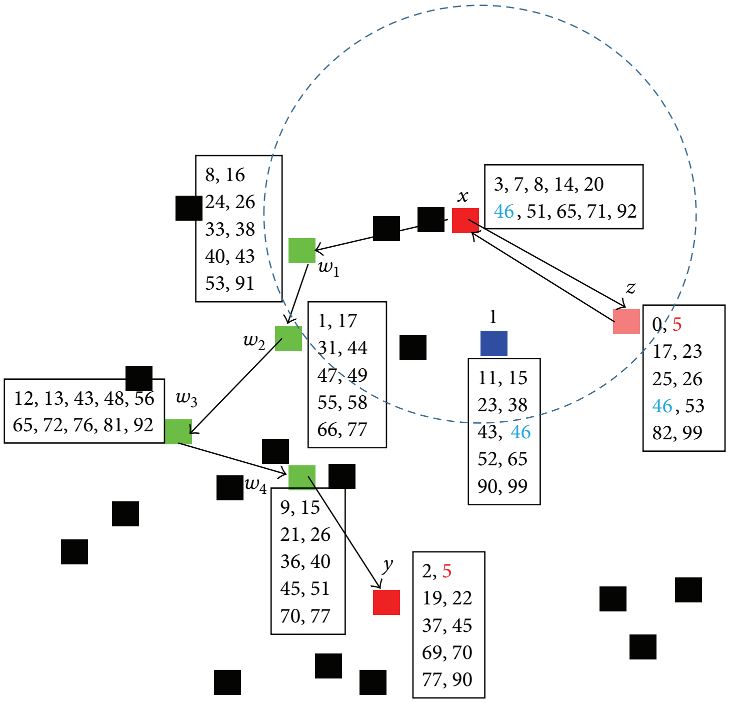


FIGURE 8: The path key is compromised in the indirect encryption. [2]x is source node and y is destination node. The attacker captures a node (the blue node) whose key ring includes key identifier 46. So the path key between x and y may be eavesdropped only near the link (x-z).

in step 4. Hence, the extra traffic is $O(\lceil \log_2 N\rceil + 2(\lceil \log_2 L\rceil + |\text{CT}|) + [(n-1)\cdot P_{xy}\cdot P_{xy}])$.

And because the probability that x and y do not share any keys is $\overline{P_{xy}}$, the average communication complexity is $O((t+1)\cdot(\lceil \log_2 L\rceil + |\text{CT}|) + \overline{P_{xy}}\cdot(\lceil \log_2 N\rceil + 2(\lceil \log_2 L\rceil + |\text{CT}|) + [(n-1)\cdot P_{xy}\cdot P_{xy}]))$.

The communication complexity of the new scheme is slightly higher than that of the basic scheme. Figure 10 shows that the difference of the communication complexity between the two schemes can be almost negligible.

(3) Computation Complexity. Let $|G|$ denote the time complexity of random sequence generation such as LFSR, and let $|E|$ represent the time complexity of an encryption (or decryption) operation.

The amount of calculation of sorting a (key identifier) set of size s is $O(s\cdot(s-1)/2\cdot\lceil \log_2 L\rceil)$, such as bubble sort algorithm or insertion sort algorithm. And the calculation amount of comparing two ordered sets of size s to find a common key identifier (binary search), called key identifier search, is $O(s\cdot\lceil \log_2 s\rceil\cdot\lceil \log_2 L\rceil)$. According to a key identifier, the amount of calculation of looking up the corresponding key in a key ring (binary search), denoted by key ring query, is $O(\lceil \log_2 s\rceil\cdot\lceil \log_2 L\rceil)$.

The average length of the neighbor list is $[(n-1)\cdot P_{xy}]$ and a node identifier has $\lceil \log_2 N\rceil$ bits, so the computation amount of deciding whether a node identifier is on a neighbor list, denoted by neighbor list query, will be $O([(n-1)\cdot P_{xy}]\cdot\lceil \log_2 N\rceil)$.

Case 1. If x and y have a shared key, only x and y are involved in establishing a path key regardless of the receiving and forwarding operations of nodes in the path. x executes a random sequence generation, a key identifier search, a key ring query, and a path key encryption in step 1, so the computation complexity of x is $O(|G| + s\cdot(s-1)/2\cdot\lceil \log_2 L\rceil + s\cdot\lceil \log_2 s\rceil\cdot\lceil \log_2 L\rceil + |E|)$.

Accordingly, y executes a key ring query and a decryption; hence, the computation complexity of y is $O(\lceil \log_2 s\rceil\cdot\lceil \log_2 L\rceil + |E|)$.

Case 2. If x and y do not have any shared key, x's neighbors are involved in finding the encryption key. $n-1$ neighbors may run neighbor list query (step 3), so the complexity is $O((n-1)\cdot[(n-1)\cdot P_{xy}]\cdot\lceil \log_2 N\rceil)$. About $[(n-1)\cdot P_{xy}]$ neighbors will run step 4 which includes a random

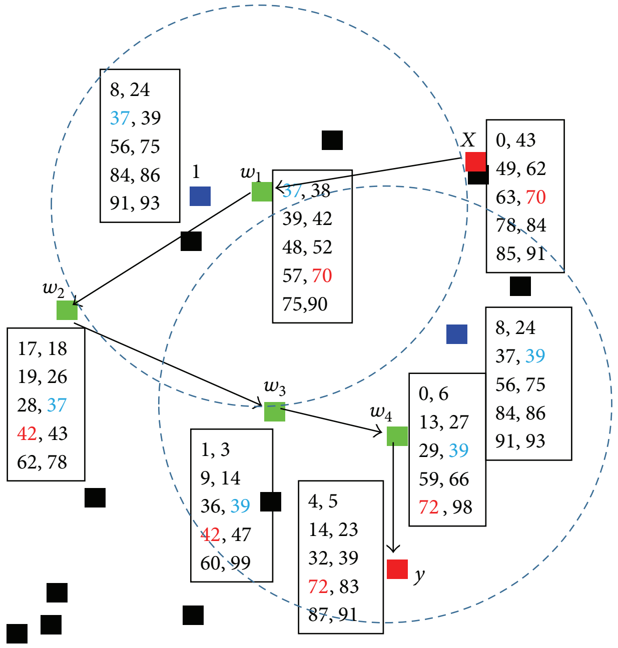


FIGURE 9: The path key is compromised in the basic scheme. ${}^3x$ is source node and y is destination node. The attacker captures a node (the blue node) whose key ring includes key identifiers 37 and 39. So the path key between x and y may be compromised at the link (w_1-w_2) which share the key 37, and it may also be compromised at the link (w_3-w_4) which share the key 39.

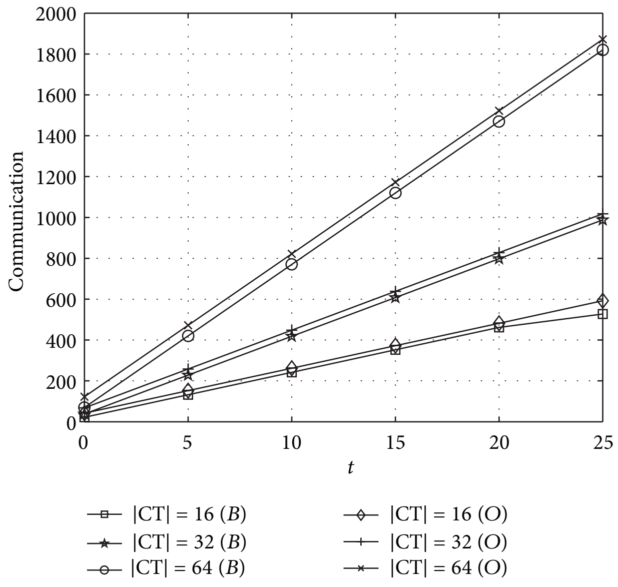


FIGURE 10: The comparison of the communication complexity.

sequence generation and a key identifier search; therefore, the computation amount is $O([(n-1)\cdot P_{xy}]\cdot(|G|+s\cdot(s-1)/2\cdot\lceil\log_2 L\rceil+s\cdot\lceil\log_2 s\rceil\cdot\lceil\log_2 L\rceil))$.

In step 5, x runs a neighbor list query and a key ring query to find the shared key with z^* and encrypts the ciphertext with the shared key. The total computation complexity is $O([(n-1)\cdot P_{xy}]\cdot\lceil\log_2 N\rceil+\lceil\log_2 s\rceil\cdot\lceil\log_2 L\rceil+|E|)$.

In step 6, z^* executes key ring query twice, encryption once, and decryption once. So the calculation amount is $O(2\lceil\log_2 s\rceil\cdot\lceil\log_2 L\rceil+2|E|)$.

Considering the above two aspects, the computational complexity is

$$\begin{aligned}
\text{Comp} &= \{\text{Comp of } x\}+\{\text{Comp of } y\}\\
&\quad+\{\text{Comp of neighbors}\}\\
&=\Bigg\{O\left(|G|+s\cdot\frac{(s-1)}{2}\cdot\lceil\log_2 L\rceil\right.\\
&\qquad\left.+s\cdot\lceil\log_2 s\rceil\cdot\lceil\log_2 L\rceil+|E|\right)\\
&\quad+\overline{P_{xy}}\cdot\Big(\left[(n-1)\cdot P_{xy}\right]\cdot\lceil\log_2 N\rceil\\
&\qquad+\lceil\log_2 s\rceil\cdot\lceil\log_2 L\rceil\Big)\Bigg\}\\
&\quad+\{\lceil\log_2 s\rceil\cdot\lceil\log_2 L\rceil+|E|\}\\
&\quad+\overline{P_{xy}}\cdot\Big\{\Big((n-1)\cdot\left[(n-1)\cdot P_{xy}\right]\cdot\lceil\log_2 N\rceil\Big)\\
&\qquad+\Big(\left[(n-1)\cdot P_{xy}\right]\\
&\qquad\cdot\Big(|G|+s\cdot\frac{(s-1)}{2}\cdot\lceil\log_2 L\rceil\\
&\qquad+s\cdot\lceil\log_2 s\rceil\cdot\lceil\log_2 L\rceil\Big)\Big)\\
&\qquad+(2\lceil\log_2 s\rceil\cdot\lceil\log_2 L\rceil+2|E|)\Big\}\\
&=O\Big(\Big(\Big(\left[(n-1)\cdot P_{xy}\right]\cdot\overline{P_{xy}}+1\Big)\cdot|G|\\
&\quad+\Big(\left[(n-1)\cdot P_{xy}\right]\cdot\overline{P_{xy}}+1\Big)\\
&\quad\cdot\left(s\cdot\frac{(s-1)}{2}\cdot\lceil\log_2 L\rceil\right)\\
&\quad+\Big(s+\left[(n-1)\cdot P_{xy}\right]\cdot\overline{P_{xy}}\cdot s+3\overline{P_{xy}}+1\Big)\\
&\quad\cdot(\lceil\log_2 s\rceil\cdot\lceil\log_2 L\rceil)+2\Big(\overline{P_{xy}}+1\Big)\cdot|E|\\
&\quad+n\cdot\left[(n-1)\cdot P_{xy}\right]\cdot\overline{P_{xy}}\cdot\lceil\log_2 N\rceil\Big).
\end{aligned}\tag{5}$$

In Phase 2 of the new scheme, the algorithm **G** is used to save on traffic at the cost of increasing the computation amount. So the total traffic in this stage is about $O(N\cdot s\cdot\lceil\log_2 L\rceil)$, and the cost of computation is about $O(N\cdot(|G|+s\cdot(s-1)/2\cdot\lceil\log_2 L\rceil+s\cdot\lceil\log_2 s\rceil\cdot\lceil\log_2 L\rceil))$. Finally, the complexity of the basic scheme and the new scheme is listed in Table 3.

The efficiency (in Phase 3) of the basic scheme and the new scheme is shown in Figures 11 and 10. In those figures, symbol (O) denotes the new scheme and symbol (B) denotes the basic scheme.

From (5), the computational complexity of the new scheme is concerned with L, s, and $|E|$. Besides that, the basic scheme varies with the number of hops t, which is shown in

Table 3: The complexity of the basic scheme and the new scheme.

	Basic scheme	New scheme
Storage	$O\left(s\cdot\lvert\text{key}\rvert+s\cdot\lceil\log_2 L\rceil+P_{xy}\cdot(n-1)\cdot(\lceil\log_2 N\rceil+\lceil\log_2 L\rceil)\right)$	$O\left(s\cdot\lvert\text{key}\rvert+s\cdot\lceil\log_2 L\rceil+P_{xy}\cdot(n-1)\cdot(\lceil\log_2 N\rceil+\lceil\log_2 L\rceil)\right)$
Communication (Phase 2)	$O\left(N\cdot s\cdot\lceil\log_2 L\rceil+N\cdot\lceil\log_2 N\rceil\right)$	$O\left(N\cdot s\cdot\lceil\log_2 L\rceil\right)$
Computation (Phase 2)	$O\left(N\cdot s\cdot\lceil\log_2 s\rceil\cdot\lceil\log_2 L\rceil\right)$	$O\left(N\cdot\left(\lvert G\rvert+s\cdot\dfrac{s-1}{2}\cdot\lceil\log_2 L\rceil+s\cdot\lceil\log_2 s\rceil\cdot\lceil\log_2 L\rceil\right)\right)$
Communication (Phase 3)	$O\left((t+1)\cdot(\lceil\log_2 L\rceil+\lvert\text{CT}\rvert)\right)$	$O\left((t+1)\cdot(\lceil\log_2 L\rceil+\lvert\text{CT}\rvert)+\overline{P_{xy}}\cdot\left(\lceil\log_2 N\rceil+2\left(\lceil\log_2 L\rceil+\lvert\text{CT}\rvert\right)+\left[(n-1)\cdot P_{xy}\cdot P_{xy}\right]\right)\right)$
Computation (Phase 3)	$O\left((t+1)\cdot 2\left((n-1)\cdot P_{xy}\cdot\lceil\log_2 N\rceil+\lceil\log_2 s\rceil\cdot\lceil\log_2 L\rceil+\lvert E\rvert\right)\right)$	Computation in formula (5)

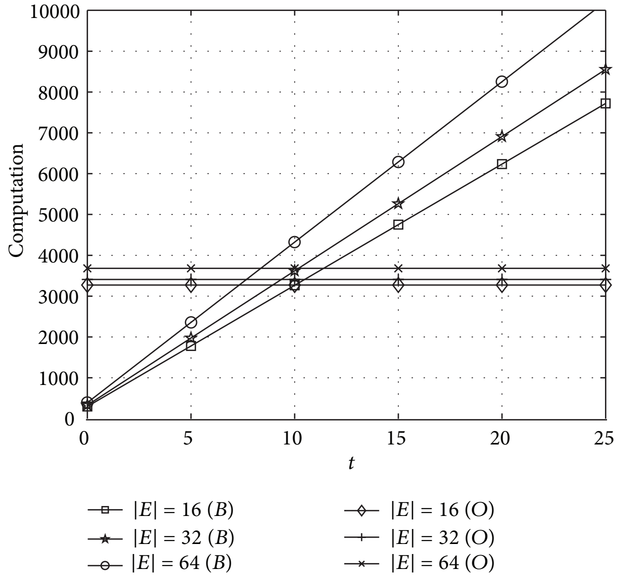


Figure 11: Comparisons of the computation complexity.

Figure 11 ($L = 100$ and $s = 10$). It can be easily noticed that if t is small, the basic scheme performs faster than the new scheme. However, the computation complexity of the basic scheme grows rapidly with the increase of the number of hops t. When t is more than 10, the complexity of the new scheme is lower than that of the basic one.

It is worth noting that the computation complexity of the basic scheme in Table 3 does not include the extra computation cost of finding logically connected neighbors in routing phase, which will be estimated in the next section.

6. Simulations

In this section, we use computer to simulate the basic scheme and our scheme. In the basic scheme, every hop of the path is not only physically but also logically connected. We try to estimate the amount of computation on deciding whether two adjacent nodes are logically connected (neighbor list query) in routing process. On the other hand, we prove that the path length between two nodes will increase in the basic scheme. At last, the execution time of the path key establishment of the new scheme and the basic scheme is tested. The simulation environment is as follows:

the operating system: ubuntu10.04,

CPU: Intel Core2 Quad Q8400 2.66 GHz,

RAM: 4.00 GB,

communication bandwidth: 4 KB/s,

routing protocol: AODV.

In those simulations, parameters are set as $N = 1000$, $L = 100$, $s = 10$, and $n = 28$.

(1) Computation Amount. As previously metioned, the basic scheme will spend lots of extra computation on neighbor list query in the routing process. Table 4 shows the average number of nodes which execute the neighbor list query in 208 random routing trails. Moreover, the hop limit is 30.

Table 4: The number of nodes involved in routing process of the basic scheme.

Number of hops	Number of trails	Average number of nodes
1–5	19	606
6–10	37	552
11–15	36	548
16–20	28	612
21–25	36	570
26–30	20	656

Obviously, more than half nodes are involved in a routing process. The computation costs on a neighbor list query are $O([(n-1) \cdot P_{xy}])$, so the total computation complexity is $O(N/2 \cdot [(n-1) \cdot P_{xy}])$. The theoretical value is more than 90000 when parameters are the same as those of routing trails.

(2) Path Length. In fact, the path length t of the basic scheme may be longer than that of the new scheme since the routing protocol is affected by the basic scheme. Consequently, the communication complexity of the basic scheme will be more than the communication amount shown in Figure 10. We randomly chose two nodes which are physically connected neighbors and then conducted 10240 routing trails to find a path in which two adjacent nodes are logically connected. The length of the path in each trail was counted, and the count results are shown in Table 5.

Obviously, the probability that the path length increases ($t > 0$) because of the influence of the basic scheme is at least $1/2$.

It should be pointed out that parameters in those simulations are small, for instance, $L = 100$ and $s = 10$, because we assume that the network has a lot of simple sensor nodes which have limited power. In fact, if parameters increase, $L = 10000$ for example, the above conclusions also hold.

(3) Execution Time of the Path Key Establishment. We simulate the path key establishment process on the computer, in which the encryption algorithm is DES. Figure 12 shows the average time spent on path key establishment.

The x-coordinate in Figure 12 is the physical hops of two nodes (x, y). As mentioned above, the real number of hops in the path key establishment of the basic scheme may be larger than that. "Direct" means that x and y share a key K_{xy}. The path key is encrypted with this key, and then pk is sent to y. "Indirect" means that x transmits the pk to y with the help of a neighbor z^*. It can be seen in the figure that our scheme is slightly more efficient than the basic scheme.

7. Conclusions

The random key predistribution scheme is a good solution to distributed sensor networks, which reduces the energy

Table 5: The growth of the number of hops.

	$t = 0$	$t = 1$	$t = 2$	$t = 3$	$t \geq 4$
Number of trails	4777	3428	1441	456	138
Percent	46.65	33.48	14.07	4.45	1.35

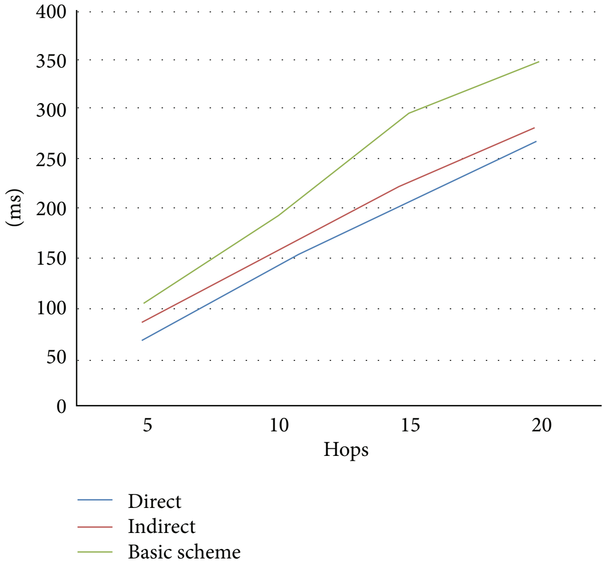


Figure 12: The simulation result of path key establishment.

consumption of nodes though losing a little connectivity and security of the scheme. In this paper, a new RKPD scheme which is independent of the routing protocol is given. The new scheme can be parameterized to meet the appropriate level of performance and keep the stability of the connectivity in a certain range by bringing small redundant in parameters.

Moreover, an algorithm with the input of node's identifier is constructed to reduce the communication overload. However, the traffic of the new scheme in the path key establishment phase is still higher than that of the basic scheme. Since Pottic and Kaiser [25] revealed that the energy of transmitting 1 K bits over 100 meters could be used to execute 3×10^6 instructions, the communication overload of the new scheme should be further reduced. A intuitive solution is that we do not generate a new path key. Instead, if x and y have a shared key k_{xy}, the key is used as the path key. Else, the key$_{k_{yz^*}}$ that z^* shares with y will be sent to x (covered by key$_{k_{xz^*}}$) by z^*. In this way, x only transmits the key identifier k_{xy} or k_{yz^*} to y. However, the above solution may incur new security flaw which is the same as the scheme in [17]. For example, a captured node may be cloned and redeployed in WSNs, and then they send fake message (as step 1 in the path key establishment) to its neighbors and defraud fresh (not compromised) keys in the key pool.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by National Natural Science Foundation of China (nos. 61370194 and 61202082) and the Fundamental Research Funds for the Central Universities (nos. BUPT2012RC0219 and BUPT2013RC0311).

References

[1] T. Matsumoto and H. Imai, "On the KEY PREDISTRIBUTION SYSTEM: a practical solution to the key distribution problem," in *Advances in Cryptology—CRYPTO'87*, vol. 293 of *Lecture Notes in Computer Science*, pp. 185–193, Santa Barbara, Calif, USA, August 1987.

[2] R. Blom, "An optimal class of symmetric key generation systems," in *Proceedings of the EUROCRYPT 84 Workshop on Advances in Cryptology*, pp. 335–338, Paris, France, April 1984.

[3] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly secure key distribution for dynamic conferences," in *Advances in Cryptology —CRYPTO' 92*, vol. 740 of *Lecture Notes in Computer Science*, pp. 471–486, 1993.

[4] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks," *ACM Transactions on Information and System Security*, vol. 8, no. 2, pp. 228–258, 2005.

[5] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security*, pp. 52–61, October 2003.

[6] D. Chakrabarti, S. Maitra, and B. Roy, "A key pre-distribution scheme for wireless sensor networks: merging blocks in combinatorial design," *International Journal of Information Security*, vol. 5, no. 2, pp. 105–114, 2006.

[7] S. Mitra, S. Mukhopadhyay, and R. Dutta, "Unconditionally-secure key pre-distribution for triangular grid based wireless sensor network," *Journal of Applied Mathematics and Computing*, vol. 44, no. 1-2, pp. 229–249, 2014.

[8] M. F. Younis, K. Ghumman, and M. Eltoweissy, "Location-aware combinatorial key management scheme for clustered sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 8, pp. 865–882, 2006.

[9] D. Liu and P. Ning, "Improving key predistribution with deployment knowledge in static sensor networks," *ACM Transactions on Sensor Networks*, vol. 1, no. 2, pp. 204–239, 2005.

[10] D. W. Carman, "New directions in sensor network key management," *International Journal of Distributed Sensor Networks*, vol. 1, no. 1, pp. 3–15, 2005.

[11] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: security protocols for sensor networks," in *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '01)*, pp. 189–199, July 2001.

[12] S. Hussain, F. Kausar, and A. Masood, "An efficient key distribution scheme for heterogeneous sensor networks," in *Proceedings of the International Wireless Communications and Mobile*

Computing Conference (IWCMC '07), pp. 388–392, August 2007.

[13] A. Rasheed and R. Mahapatra, "Key predistribution schemes for establishing pairwise keys with a mobile sink in sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 1, pp. 176–184, 2011.

[14] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS '02)*, pp. 41–47, November 2002.

[15] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of the IEEE Symposium on Research in Security And Privacy*, pp. 197–213, May 2003.

[16] T. Shan and C. Liu, "Enhancing the key pre-distribution scheme on wireless sensor networks," in *Proceedings of the IEEE Asia-Pacific Services Computing Conference*, pp. 1127–1131, 2008.

[17] C.-F. Law, K.-S. Hung, and Y.-K. Kwok, "A novel key redistribution scheme for wireless sensor networks," in *Proceedings of the IEEE International Conference on Communications (ICC '07)*, pp. 3437–3442, June 2007.

[18] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Establishing pair-wise keys for secure communication in ad hoc networks: a probabilistic approach," in *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP '03)*, vol. 1, pp. 326–335, 2003.

[19] R. Di Pietro, L. V. Mancini, and A. Mei, "Random key-assignment for secure wireless sensor networks," in *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor networks (SASN '03)*, pp. 62–71, October 2003.

[20] W. Gu, X. Bai, S. Chellappan, and D. Xuan, "Network decoupling for secure communications in wireless sensor networks," in *Proceedings of the 14th IEEE International Workshop on Quality of Service (IWQoS '06)*, pp. 188–198, June 2006.

[21] W.-S. Li, T.-S. Su, and W.-S. Hsieh, "Multi-neighbor random key pre-distribution: a probabilistic analysis," *IEEE Communications Letters*, vol. 13, no. 5, pp. 306–308, 2009.

[22] M. Abazeed, N. Faisal, S. Zubair, and A. Ali, "Routing protocols for wireless multimedia sensor network: a survey," *Journal of Sensors*, vol. 2013, Article ID 469824, 11 pages, 2013.

[23] http://en.wikipedia.org/wiki/Linear_feedback_shift_register.

[24] D. H. Yum and P. J. Lee, "Exact formulae for resilience in random key predistribution schemes," *IEEE Transactions on Wireless Communications*, vol. 11, no. 5, pp. 1638–1642, 2012.

[25] G. Pottic and W. Kaiser, "Wireless sensor networks," *Communications of the ACM*, vol. 43, pp. S1–S8, 2000.

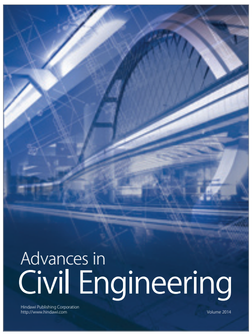
Journal of
Engineering

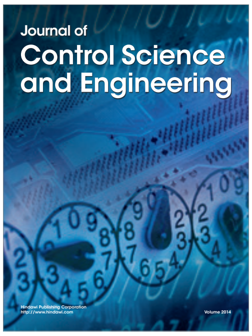
The Scientific World Journal

International Journal of
Rotating Machinery

Journal of
Sensors

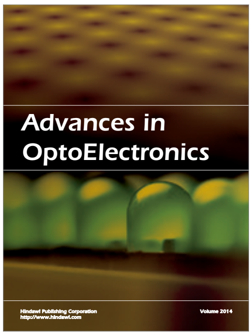
International Journal of
Distributed Sensor Networks


Advances in
Civil Engineering


Journal of
Control Science and Engineering

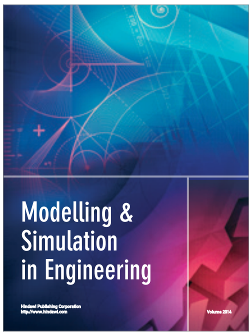
Journal of
Robotics

Journal of
Electrical and Computer Engineering

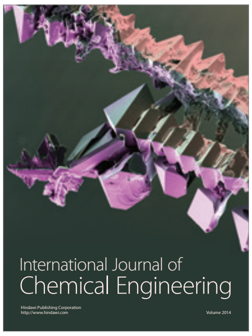

Advances in
OptoElectronics

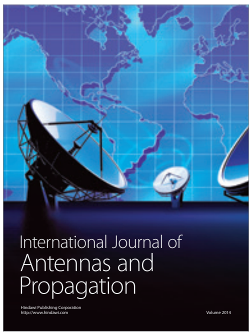
VLSI Design

International Journal of
Navigation and Observation

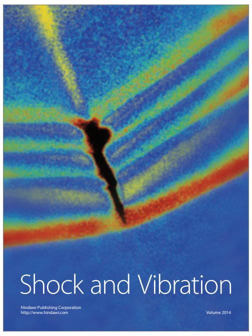

Modelling & Simulation in Engineering

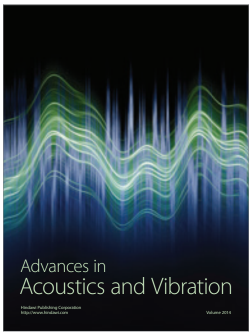
International Journal of
Aerospace Engineering


International Journal of
Chemical Engineering


International Journal of
Antennas and Propagation

Active and Passive Electronic Components


Shock and Vibration


Advances in
Acoustics and Vibration