PAPER IEICE/IEEE Joint Special Section on Autonomous Decentralized Systems Technologies and Their Application to Networked Systems

Improvement of Dependability against Node Capture Attacks for Wireless Sensor Networks**

Eitaro KOHNO†*, **Tomoyuki OHTA**†, **Yoshiaki KAKUDA**†, ***and*** **Masaki AIDA**††, ***Members***

SUMMARY A Wireless Sensor Network has sensor nodes which have limited computational power and memory size. Due to the nature of the network, the data is vulnerable to attacks. Thus, maintaining confidentiality is an important issue. To compensate for this problem, there are many countermeasures which utilize common or public key cryptosystems that have been proposed. However, these methods have problems with establishing keys between the source and the destination nodes. When these two nodes try to establish new keys, they must exchange information several times. Also, the routes of the Wireless Sensor Networks can change frequently due to an unstable wireless connection and batteries running out on sensor nodes. These problems of security and failure become more serious as the number of nodes in the network increases. In this paper, we propose a new data distribution method to compensate for vulnerability and failure based on the Secret Sharing Scheme. In addition, we will confirm the effect of our method through experiments. Concerning security, we compare our method with the existing TinySec, which is the major security architecture of Wireless Sensor Networks.

key words: *wireless sensor networks, security, node capture attack, secret sharing scheme, key refreshment*

1. Introduction

Highly confidential information relating to fields such as crime prevention, healthcare, disaster prevention, and so on, is often sent across Wireless Sensor Networks (WSNs) [1]. In such cases, it is important to prevent attacks such as eavesdropping, unauthorized packet insertion, and data compromise. Wireless data transfer uses WSNs, and the sensor nodes of WSNs have strong limitations on computational resources and memory size. Another problem facing the node is the energy constraint, as the batteries can run out easily. Some countermeasures against this have been reported [2]. Among them [3] is a method based on fault-tolerant techniques. In this method, the original data is duplicated and transmitted using multiple paths to the sink node. Whenever some intermediate node exhausts its battery, the sink node can receive data using the voting technique.

Manuscript received April 26, 2010.

Manuscript revised August 26, 2010.

†The authors are with the Graduate School of Information Sciences, Hiroshima City University, Hiroshima-shi, 731–3194 Japan.

††The author is with the Graduate School of System Design, Tokyo Metropolitan University, Hino-shi, 191–0065 Japan.

*Presently, with the Graduate School of System Design, Tokyo Metropolitan University.

**This work is based on "Secure decentralized data transfer against node capture attacks for wireless sensor networks," by E.Kohno, T.Ohta, and Y.Kakuda which appeared in Proc. IEEE International Symposium on Autonomous Decentralized Systems (ISADS 2009), Athens Greece, March 2009, ©IEEE

DOI: 10.1587/transinf.E94.D.19

TinySec [4] is one of the most popular security architectures for keeping data confidential. TinySec provides mechanisms of encryption and authentication to WSNs using the common key cryptosystems. This method allows us to overcome eavesdropping attacks and authenticate both nodes. However, using a key creates a new problem: all common and public key cryptosystems have a secret key which must be protected. One of the most popular key protection methods was first proposed by Eschenauser and Gligor [5]. Their main idea is based on randomly predistributing a key to each node. When two nodes want to communicate with each other, they will search for a key to share. Their method is known as the random key pre-distribution scheme (RKP). RKP has been improved by their successors [6], [7]. However, their method becomes invalid when the keys are compromised.

The pairwise key pre-distribution scheme [8]–[10] is one of the key pre-distribution schemes proposed to improve RKP's shortcomings. "TinyKeyMan" is one example that implements the pairwise key pre-distribution scheme, using a pool of randomly generated bi-variate t-degree polynomials to generate keys. It evaluates the effect of compromised nodes on WSNs. The main disadvantage of the pairwise key pre-distribution scheme is its complexity, which makes it hard to implement and increases overhead costs. In addition, the pairwise key pre-distribution scheme is considered neither key revocation nor key refreshment. The pairwise key pre-distribution scheme, like its predecessors TinySec and RKP, becomes invalid when keys are compromised.

The above-mentioned methods become invalid when the keys are compromised. To deal with this drawback, a method of scrambling program codes and keys [11] using code obfuscation techniques [12] was proposed. This method requires more memory and execution time than TinySec. Furthermore, the key sharing system generates overhead both in time required to share new keys and transmit control messages. During key establishment, key sharing, and key refreshment, the control packets are exchanged several times. These are the main obstacles that must be overcome.

In this paper, we propose a new distribution method that utilizes the Secret Sharing Scheme and is resilient against node capture attacks. In addition, we confirm the ability of our method to improve resiliency against node capture attacks, compared with TinySec. Also we compare the agreement time of our scheme to existing methods.

The rest of the paper is organized as follows: in Sect. 2,

we introduce node capture attacks and some problems with wireless sensor networks. In Sects. 3 and 4, we highlight the secret sharing scheme and the proposed scheme, respectively. In Sect. 5, we discuss the performance of our scheme against node capture attacks. We present our conclusion on the proposed method in Sect. 6.

2. Fault Detection and Security in WSNs

2.1 Mica Mote

A Mote [13] is one of the most popular implementations of sensor nodes in WSNs. The platform of the Mote varies depending on the type of hardware and the communication device. Table 1 shows the most popular platforms and specifications of the Mote. Each platform has very limited computational power and memory size compared to a PC. MICAz and MICA2 have the same specifications except those depending on the communication device.

The Mote has a special operating system for embedded devices, called TinyOS [14]. The development environment and tools are distributed as open source software. In addition, we use the simulator TOSSIM [15] to check the behavior on a real machine. TOSSIM includes some supplemental tools such as LossyBuilder, and so on.

2.2 Threat Model of Node Capture Attacks

The nodes in WSNs use radio wave links to communicate with each other. In addition, sensor nodes need to be exposed to the environment for a long time to take measurements. As a result, WSNs may be affected by many kinds of attacks. This section describes the attacks which are targeted in this paper.

Zhang [16] et al. list three kinds of attacks on wireless networks: eavesdropping, compromising, and node insertion. In this paper, a node capture attack is defined as keys/data inside a node being compromised or stolen. The eavesdropping attack can be prevented by various cryptographic methods using a common (secret) key. TinySec is one of the most popular architectures of these methods.

Table 1 Platforms of Mote and their hardware specifications.

Platform	MICAz	MICA2	MICA
CPU	ATmega 128		ATmega 103
Clock (MHz)	7.37	7.37	4
Program Memory (kByte)	128	128	128
SRAM (kByte)	4	4	4
Radio Frequency (MHz)	2405	315/433 /915	433/915
Maximum Data Rate (kbps)	250	38.4	40

However, systems such as TinySec are comparatively weak against node compromising. Once the keys are compromised, they become invalid, and the system is corrupted.

3. Secret Sharing Scheme

The Secret Sharing Scheme (SSS) [17]–[19] was proposed by Shamir [18] and Blakley [19] independently in 1979. Simmons [17] summarized the findings of their methods in the literature. In Shamir's method, SSS was realized by the solution of a polynomial. When we encrypt the original information, say S, n pieces of data will be created from S. Each piece of data is called a "share." In Shamir's SSS, we can decrypt the original data by collecting k number of shares. If the number of collected data is less than k, we can not recover the original data. Therefore, k is called the threshold number, where n and k are positive integers and n must be greater than or equal to k. Shamir's SSS is also called a (k, n) threshold scheme. A detailed explanation of Shamir's method is as follows: at the encryption, we calculate shares using the following Eq. (1):

$$f(x) = S + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1} \tag{1}$$

where a_i $(i = 1, 2, \cdots, k-1)$ are random integers. We obtain each share (u_i, v_i) $(i = 1, 2, \cdots, n)$ by substitution of the value u_i $(i = 1, 2, \cdots, n)$ for $f(x)$. The calculations of Eq. (1) are performed over the prime field $GF(q)$, where q is a sufficiently large prime number.

When we decrypt the original data, we need a set of at least k shares to calculate. We can decrypt the original data using Lagrange's interpolation method as in Eqs. (2) and (3).

$$S = \lambda_1 v_1 + \lambda_2 v_2 + \cdots + \lambda_k v_k \tag{2}$$

where

$$\lambda_j = \prod_{i=1, i\neq j}^{k} \frac{u_i}{(u_i - u_j)}. \tag{3}$$

Equations (2) and (3) are also calculated over $GF(q)$. In Shamir's scheme, $q \geq \max(n, S)$. For Shamir's scheme, Karnin [20] proposed and discussed the calculation equations from (1) to (3) over the extension field $GF(q^m)$. In Karnin's method, $n \leq q^m + k - 2$. In Karnin's proposal, calculations may be made more easily than in Shamir's method, because we can transform the addition and the subtraction between two numbers into XOR of each element of the number. Also, the multiplication and the division become addition and subtraction over $GF(q)$ of the primitive element. For the proposed method, we employ Karnin's method. The threshold scheme uses redundancy to protect the original data from being compromised or lost.

The conceptual diagram of the SSS is illustrated in Figs. 1 and 2.

If the collected number of shares is less than k, we cannot obtain the original data with Eqs. (2) and (3).

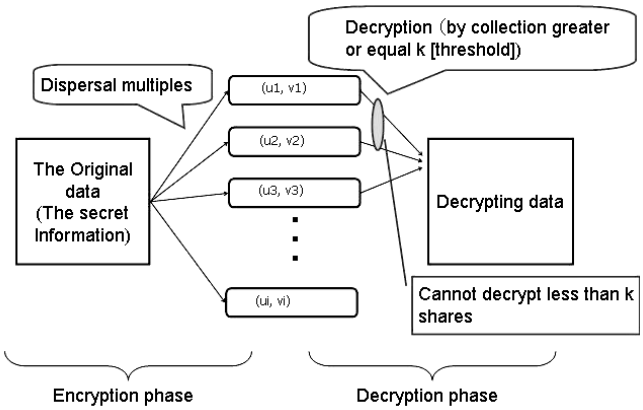

Fig. 1 Conceptual diagram of secret sharing scheme.

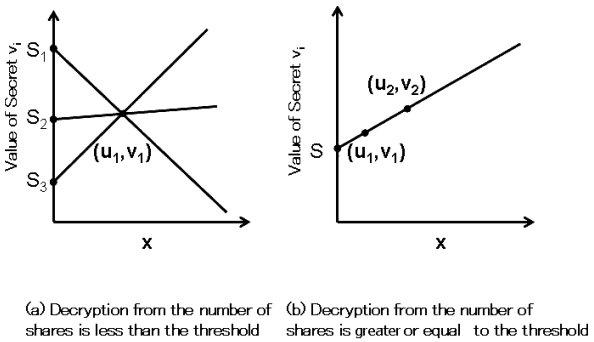

Fig. 2 Decryption of secret information on a $(2, n)$ threshold scheme.

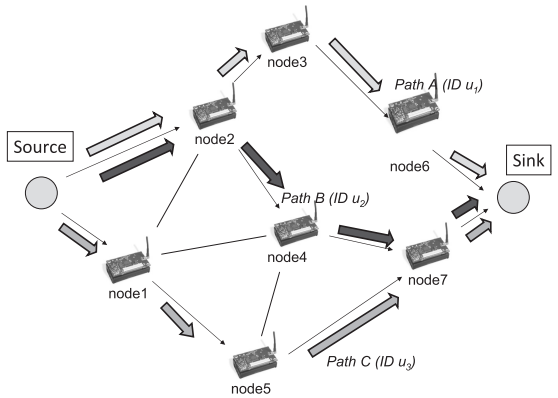

Fig. 3 Assumptions of the proposed method.

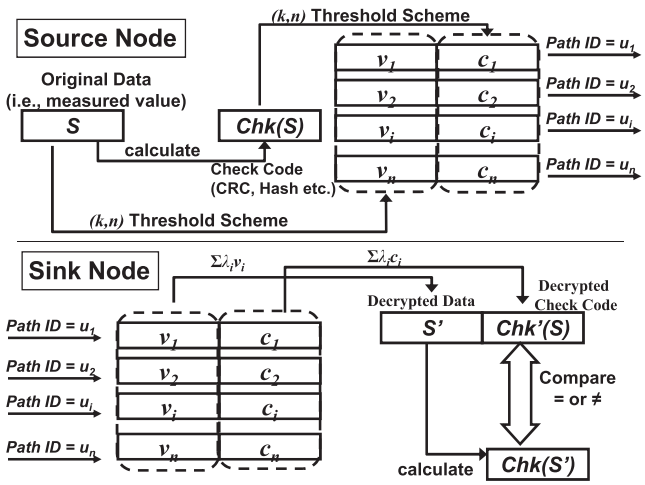

Fig. 4 Conceptual diagram of our proposed method.

4. Proposed Method

4.1 Assumptions

We make the following assumptions:

- Each node which transmits measured data has one or more paths to the sink node.
- If the node has multiple paths to the sink, the node can distinguish each path with an identifier.
- All of the nodes pre-share the same irreducible polynomial $p(x)$.

Figure 3 depicts an example network which reflects our assumption. It shows a node which has three paths to the sink node. Each path can be distinguished by $ID = u_1, u_2, u_3$. While the method of finding multiple paths is outside the scope of this paper, many multi-path routing methods exist.

4.2 Overview and Key Idea of Proposed Method

The conceptual diagram of our proposed method is illustrated in Fig. 4. Our basic idea is to transmit an encrypted check code $Chk(S)$ while employing the threshold scheme on a sensor node; the sensor node calculates the check code (*e.g.* check sum, hash without key, and so on) before dispersing the original data. Afterwards, the sensor node individually calculates each share by threshold scheme from the original data and the check code value. The sink node then compares the recovered data S' from each share to the decrypted check code $Chk'(S)$.

Suppose that the calculated share from the check code is denoted as c_i. Each share (u_i, v_i, c_i) is transmitted along an appropriate path. When each share reaches the sink node, it is decrypted using Eqs. (2) and (3). At this time, the source node does not seek agreement on the threshold value with the sink node. In addition, the sensor node (the source node) does not agree with the sink node on the threshold value. This means the source node can change the threshold value at any time without having to consult with the sink node.

In a WSN such as in Fig. 3, we consider two types of emergent events: a node fault and a node capture. If a node fault occurs, the data cannot be transmitted to any of its neighboring nodes. On the other hand, if a node is captured by an adversary, it can be manipulated in any number of ways. In this paper, we assume that the captured node fabricates the transmitted data. As a result, WSNs are classified into 4 cases: (a) without any event, (b) with one or more node faults, (c) with one or more node captures, (d) with no event other than re-routing.

According to decryption by our method, case (a) is identified if the data is decrypted correctly; cases (b) and (c) have occurred if it fails the verification by check code after being decrypted, and case (d) occurs if the data is decrypted correctly but the threshold number does not match.

4.3 Characteristics of the Proposed Method

On shared secret-key or public-key based cryptosystems, the system needs to refresh keys when it detects compromised keys and so on. When the system changes the keys of two nodes (*i.e.*, between the source and the destination nodes), the nodes must exchange the information a number of times to create new keys. As a result, the length of the hop count increases in key based systems. However, our system can change the threshold number and the polynomial $f(x)$ without needing to exchange information: therefore the length of hop counts is unaffected. This is a significant advantage over the key based systems.

4.4 Algorithm

The following data existed in collaborating nodes:

- Source nodes:
  - Threshold value k,
  - Coefficients a_i of polynomial of SSS,
  - Path IDs u_i,
  - Original data S,
  - Check node value $Chk(S)$,
  - Shares to transfer $(u_i.v_i, c_i)$.
- Sink node
  - Each share (u_j, v_j, c_j),
  - Previous threshold value k',
  - Decrypted data S' from (u_j, v_j),
  - Check code value $Chk'(S)$ from (u_j, c_j).
- Intermediate relay nodes
  - Shares to transfer $(u_i.v_i, c_i)$.

Following procedures are performed in collaborating nodes:

- **Procedure of source node**
  First of all, we calculate the check code values $Chk(S)$ from the original data S. When each source node calculates shares to fit the number of paths to the sink node, the node generates random numbers for $a_i (i = 1, 2, \cdots, k-1)$. We employ a (k, n) threshold scheme based on Karnin's method to produce n shares, where $n - 1 \geq k \geq 2$ except for cases where the minimum hop count between the source and the destination node is one. The threshold value of k can be set by each source node individually, without needing to consult the sink node. Also, the threshold value can change each time in the same node.

Table 2 Comparison of memory size for various platforms and methods.

Platform	Memory	TinyOS	TinySec	Proposed
MICA	ROM (kByte)	8.21	18.4 (2.24)	12.3 (1.50)
	RAM (Byte)	336	616 (1.83)	935 (2.78)
MICA2	ROM (kByte)	10.5	19.4 (1.85)	14.5 (1.38)
	RAM (Byte)	447	706 (1.58)	1044 (2.34)
MICAz	ROM (kByte)	9.87	- (-)	13.9 (1.40)
	RAM (Byte)	392	- (-)	991 (2.53)

- **Procedure of sink node**
  The sink node uses Eqs. (2) and (3) to calculate the original data and its check code $Chk(S')$ value using the share from the source node. The accuracy of the decrypted data is checked by the confirmation of its decrypted check code value, $Chk'(S)$.

4.5 Implementation of Proposed Method

Using the proposed method, we implemented the prototype system on the TinyOS 1.15 with nesC, a compiler. Table 2 shows the comparison of memory size in the implementation of the proposed method. On implementation, we use $GF(2^8)$ and $q(x) = x^8 + x^7 + x^2 + x + 1$ for the threshold scheme, and the CRC-16 function from the ITU-T CRC standard with $g_{16}(x) = x^{16} + x^{12} + x^5 + 1$ [21]. In Table 2, "TinyOS" does not include security support, "TinySec" shows TinySec with Skipjack, and "Proposed" shows our proposed method. In Table 2, "-" shows no data, because we cannot compile the prototype program. Table 2 shows, for each of the three methods, memory size is within the acceptable range of RAM and ROM. The program sizes of the prototype systems include the encryption procedures on the sender node and the decryption procedures on the sink node. The number in parentheses shows the ratio of the values to its "TinyOS" values. The ROM size of our method is smaller than that of "TinySec," because the calculation of the solutions of (1),(2), and (3) is more lightweight than the key-based cryptosystems. However, the RAM size of our method is larger than that of "TinySec." It is due in large part to the buffer size needed to store each share until finishing the SSS calculation for decryption on the sink node.

5. Simulation Results and Discussion

5.1 Experiment for Evaluation

We define WSNs security in terms of its resiliency against single node capture attacks. We determine the reliability of WSNs by the expected number of compromised nodes,

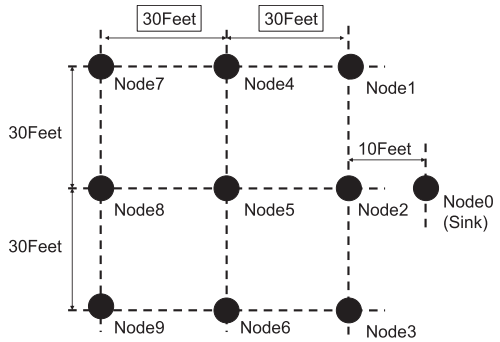

Fig. 5 The topology of the experiment.

which is equivalent to stolen data. In addition, we also evaluate execution time and total size of the transferred packets, which are the overhead of the system. For the experiment, we used the topology illustrated in Fig. 5. The network has a lattice consisting of one sink node and nine source nodes. The packet loss rate of the system was generated by LossyBuilder, to reflect the results of MICA2. In the experiment, the multiple paths from the source node to the sink were determined automatically from the packet loss rate. The path does not include any link with the packet loss rate being 5 % or above. When the packet loss rate of a link is 5 % or above, the link is assumed to be disconnected. We set up the route from each source node to the sink with static routing. In the (k, n) threshold scheme, we employ the value of threshold of $k = n - 1$. When $n = 2$, however, we employ the value of threshold of $k = 2$. That is, we employ (2, 2) threshold scheme. We have eliminated the capture of the sink node by a third party. For the preliminary experimentation, we perform simulation as $n = k$ for Sects. 5.1 and 5.2. The consideration of the value k our scheme is mentioned in Sect. 5.3.

The simulated attack was carried out on nodes as follows: during the transmission from the source nodes, all sensor nodes (including the source nodes and the intermediate relay nodes) record the relayed data. After every transmission, we decrypt the data using relayed shares in each node. Each node is treated as a possibly captured node. To simulate a captured node, we embed the decryption algorithms into the memory of each node including the sink node. That is, every sensor node uses Eqs. (2) and (3) to decrypt the data and calculate its check code $Chk(S')$ value using the shares which are stored in the nodes. When the decrypted check code value $Chk'(S)$ of a node is equal to $Chk(S')$, the original data can be stolen by node capture.

In defending against attacks, following resources or data is compromised:

- Source nodes:
  - Threshold value k,
  - Coefficients a_i of polynomial of SSS,
  - Path IDs u_i,
  - Original data S,
  - Check node value $Chk(S)$,
  - Shares to transfer $(u_i.v_i, c_i)$.
- Intermediate nodes:
  - Shares to transfer $(u_i.v_i, c_i)$.

Table 3 Results of experiments.

Items	TinyOS	TinySec	Proposed
Number of data streams when a single node was compromised	1.87	1.87 (1.00)	1.51 (0.81)
Execution time (sec)	1.36	1.40 (1.03)	4.20 (3.08)
Amount of data (Byte)	36.0	41.0 (1.14)	124 (3.44)

Table 4 Overhead comparison of modified and proposed methods.

Items	Modified	Proposed
Number of data streams when a single node was compromised	1.51 (0.81)	1.51 (0.81)
Execution time (sec)	3.05 (2.24)	4.20 (3.08)
Amount of data (Byte)	90.2 (2.51)	124 (3.44)
ROM (kByte)	12.3 (1.14)	12.7 (1.17)
RAM (Byte)	505 (1.13)	521 (1.18)

In our experiments, source and intermediate nodes were captured. When a source node was captured, the original data S was stolen by an adversary. When an intermediate node was captured, an adversary could be steal shares which were relayed.

Table 3 shows the results of the experiment. The data in Table 3 are the average of 10 trials. The bracketed numbers in Table 3 indicate an increasing ratio based on each value of its "TinyOS." As seen in Table 3, there is an increase in the ratio of execution time and amount of data (the number of packets) in the proposed method. Next, we consider the relation between the shortest number of hops and reliability. Figure 6 shows the relationship of the ratio of reliability versus the shortest hops. It also indicates that our proposed method becomes more effective than TinyOS and TinySec as the number of hops increases.

5.2 Overhead

Now we will look at the overhead of nodes that are one hop count to the sink. We have modified the proposed method so as not to use it on a node that has one hop to the sink. Table 4 shows the results of our observed data. In Table 4, "Modified" refers to the modified method mentioned above; "Proposed" refers to the original proposed method.

The results of Table 4 indicate that our "Modified" method can reduce the overhead by decreasing the execution time, amount of packets, and amount of memory, without performance degradation on the resiliency (of single node capture attacks).

Table 5 Effects of threshold k value in the three conditions.

Items	TinyOS and TinySec	Always $k = n - 1$	Our method	Always $k = n$
Number of data streams when a single node was compromised	1.78	1.79 (1.01)	1.48 (0.83)	1.3 (0.73)

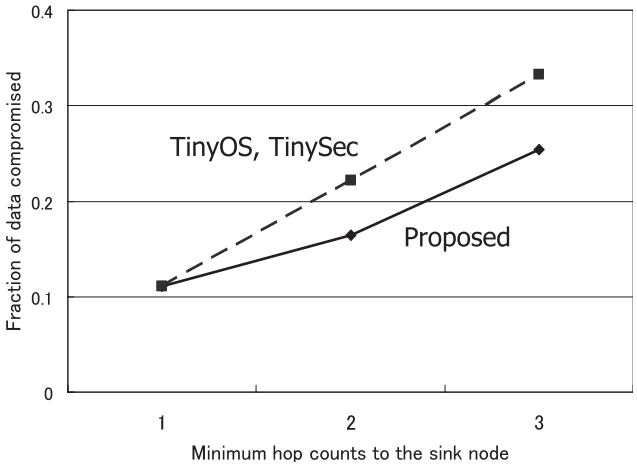

Fig. 6 Ratio of reliability and shortest hops from nodes to sink.

5.3 Effect of the Threshold Value k

In our proposed method, the value k of the (k, n) threshold method has an important role. When we employ the (k, n) threshold scheme under the condition of $k = n$, the scheme has resiliency against a node capture attack but does not have resiliency against a node failure. When we employ the (k, n) threshold scheme under the condition of $k \leq n - 1$, the scheme has resiliency against a node capture attack and a node failure. In our proposed method, we assume that $k = 2$ under the condition of $n = 2$ and $k \leq n - 1$ under the condition of $n \geq 3$. To confirm the effect of the parameter, we have performed the simulation experiments under the following three conditions:

- Always: $k = n$ and no exception
- Always: $k = n - 1$ and no exception
- Our method: $k \leq n - 1$ when $n \geq 3$, and $k = 2$ when $n = 2$.

For our experiments, we use the network topology shown in Fig. 5 under the same condition.

Table 5 shows the results of the experiments. The data in Table 5 are the average of 10 trials. In Table 5, "TinyOS and TinySec" refers to the data of the TinyOS and TinySec; "Always $k = n - 1$" refers to the data of the (k, n) threshold scheme with the condition of $k = n - 1$; "Always $k = n$" refers to the data of (n, n) threshold scheme; "Our method" refers to the data of our proposed method.

As Table 5 shows, we found that "Always $k = n$" has the best resiliency against node capture attacks, however, it has no resiliency against node faults. Thus, if we employ "Always $k = n$" for the proposed scheme, we have to recover node faults using re-routing when node faults occur. On the contrary, if we employ "Always $k = n-1$" which has resiliency against node faults, the proposed scheme has almost the same resiliency against node capture attacks. This is because, when a node's n value (*i.e.*, the number of paths) is to small such as 2, the method with the $(n-1, n)$ threshold scheme creates a new vulnerability to node capture attacks. As a result, the performance advantage of the (k, n) threshold scheme is almost cancelled out by the above-mentioned disadvantage. In our method, we can adjust the threshold value k depending on the number of paths, n. When the value of n is large, we set the value of k to $n - 1$. When the value of n is 2, we set the value of k to 2. A node fault is indicated by decryption failure, as described in Sect. 4.2. At this time, the sink node requests node recovery means such as re-transmission or re-routing.

5.4 Comparison of Resiliency of Our Method with Pairwise Key Pre-Distribution Scheme

The pairwise key pre-distribution scheme has an advantage over RKP when it comes to protection against node compromise. According to Liu et al. [8], the probability of any (direct and indirect) key between two previously non-compromised nodes being compromised, P_c, can be estimated by

$$P_c = p \times P_{cd} + (1 - p)[1 - (1 - p_c)(1 - P_{cd})^2]. \quad (4)$$

Here P_{cd} is the probability of each direct key being compromised and $1 - (1 - p_c)(1 - P_{cd})^2$ is the probability of an indirect key being compromised. In addition, $p_c = N_c/N$, where N_c is the number of compromised nodes, N is the total number of nodes in the WSN, and p is the probability of direct key establishment. Additionally, the authors assume that each pair of nodes can establish a direct or indirect key and that the network is fully connected (*i.e.*, all nodes can communicate with each other). Furthermore, P_{cd} is the probability of each direct key being compromised. P_{cd} can be estimated by

$$P_{cd} = 1 - \sum_{i=0}^{t} P(i), \quad (5)$$

where $P(i)$ is the probability of the polynomial being chosen exactly i times among N_c compromised sensor nodes.

On the other hand, in our system, the probability of data being compromised, P, can be estimated by

$$P = \frac{N_{proposed}}{N_{combination}}. \quad (6)$$

$N_{proposed}$ is the number of combinations of nodes that collect all dispersed shares of each node to the sink node in our proposed method, and $N_{combination}$ is the number of combinations of nodes compromised. $N_{combination}$ is calculated as follows:

$$N_{combination} = \sum_{l=0}^{N} {}_nC_l = \sum_{l=0}^{N} \{\frac{N!}{l!(N-l)!}\}, \quad (7)$$

where N is the total number of sensor nodes and l is the number of simultaneously compromised nodes. Also, $N_{proposed}$ is determined by network topology and the number of multiple path possibilities.

As seen in Eqs. (4) and (5), P_c tends to be smaller as the threshold value t of bi-variate polynomials increases in the pairwise key pre-distribution scheme. Meanwhile, when the WSN has many joint nodes on the paths except source nodes and the sink node, the network is vulnerable in our proposed method. While routing protocol to fit our method is outside the scope of this paper, we would like to discuss it in another paper.

5.5 Suitable Scenarios for Our Proposed Method and the Limitation in Current Protocol

Our proposed technique will be suitable in scenarios where WSNs measure emergent such as that which accompanies disasters. In this case, the information allows the public to respond. In general, although a sensor node cannot mount human sensor devices, an adversary can access the sensor. When a sensor node sends data to the sink node, dispersed data transfer along multiple paths can prevent data/keys from being compromised. Also, our proposed method verifies the decrypted data on the sink node. Since wireless links of WSNs are unstable, paths between source (sensor) nodes and the sink node can change. Generally, it is suggested that keys that part of key-based systems are refreshed periodically. However, the key sharing system generates overhead both in time required to share new keys and to transmit control messages. In contrast, our proposed method can refresh the way of encryption when a source node changes polynomial functions and the threshold value without agreement with the sink node.

In general, the current routing protocols of WSNs are single path routing protocols. That protocols are limited by path possibilities. In contrast, our proposed method is limited by bandwidth and node dencities. The bandwidth and the limitation of path possibility are the limitations in current protocol in relation to proposed technique which is limiting its complete benefit. In our proposed method, we made an assumption that there are multiple paths between the source and destination nodes on a network. When there is only one path between the source and destination nodes, our method cannot transmit dispersed shares. In other words, when the network cannot find multiple paths, our method is not effective. Our method is weak against the combination of two collaboration attacks: routing attacks and node capture attacks.

6. Conclusion

In many cases, the sensor nodes are exposed to the environment for a long time. Therefore, node capture attacks associated with stealing keys/data are important to take into account when considering network security.

In this paper, we proposed a new method resilient to node capture attacks. Our method utilizes SSS to disperse confidential information without the need of a secret key. This method was implemented on the Mote nodes. In addition, we performed simulations with TOSSIM. From the experiments, we confirmed that our method is more effective than the existing TinySec system. Additionally, we found our method tends to be more effective as the number of hops-to-sink-node increases. On the other hand, we observed an increased overhead on shorter hop nodes. We have also shown a countermeasure capable of reducing excess dispersals without degrading the resilience against node capture attacks. Furthermore we have shown the effectiveness of changing the threshold value, k, in correlation to the number of paths.

We plan to implement a routing algorithm to take advantage of our method.

Acknowledgements

This research is supported by the Ministry of Education, Science, Sports and Culture of Japan under Grant-in-Aid for Scientific Research (B) (No.21300028). This research is also supported by the National Institute of Information and Communications Technology, Japan under Early-concept Grants for Exploratory Research on New-generation Network (No.145-9) , and the Ministry of Education, Science, Sports and Culture of Japan under Grant-in-Aid for Scientific Research (C) (No.22500065).

References

[1] I. Stojmenovi'c, ed., Handbook of Sensor Networks: Algorithms and Architectures, Wiley, 2005.
[2] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," Comput. Netw., vol.38, no.4, pp.393–422, March 2002.
[3] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-resilient, energy-efficient multipath routing in wireless sensor networks," ACM SIGMOBILE Mobile Computing and Communications Review, vol.5, no.4, pp.11–25, Oct. 2001.
[4] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A link layer security architecture for wireless sensor networks," SenSys '04: Proc. 2nd International conference on Embedded networked sensor systems, pp.162–175, ACM, New York, NY, USA, 2004.
[5] L. Eschenauer and V.D. Gligor, "A key-management scheme for distributed sensor networks," CCS '02: Proc. 9th ACM Conference on Computer and Communications Security, pp.41–47, ACM, New York, NY, USA, 2002.
[6] A.C. Chan, "Probabilistic distributed key pre-distribution for mobile ad hoc networks," IEEE International Conference on Commununications (ICC 2004), pp.3743–3747, Paris, France, June 2004.
[7] H. Chan, A. Perrig, and D. Song, "Random key predistribution

schemes for sensor networks," IEEE Symposium on Security and Privacy, pp.197–213, Oakland, California, USA, May 2003.
[8] D. Liu, P. Ning, and R. Li, "Establishing pairwise keys in distributed sensor networks," ACM Trans. Inf. Syst. Secur., vol.8, no.1, pp.41–77, 2005.
[9] W. Du, J. Deng, Y.S. Han, and P.K. Varshney, "A pairwise key predistribution scheme for wireless sensor networks," CCS '03: Proc. 10th ACM Conference on Computer and Communications Security, pp.42–51, ACM, New York, NY, USA, 2003.
[10] 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 Trans. Inf. Syst. Secur., vol.8, no.2, pp.228–258, 2005.
[11] A. Alarifi and W. Du, "Diversify sensor nodes to improve resilience against node compromise," SASN '06: Proc. Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks, pp.101–112, ACM, New York, NY, USA, 2006.
[12] C. Linn and S. Debray, "Obfuscation of executable code to improve resistance to static disassembly," CCS '03: Proc. 10th ACM Conference on Computer and Communications Security, pp.290–299, ACM, New York, NY, USA, 2003.
[13] M. Horton, D. Culler, K. Pister, J. Hill, R. Szewczyk, and A. Woo, "MICA, the commercialization of microsensor motes," Sensors, vol.19, no.4, pp.40–48, April 2002.
[14] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," ACM SIGPLAN Notes, vol.35, no.11, pp.93–104, 2000.
[15] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire tinyos applications," SenSys '03: Proc. 1st International Conference on Embedded Networked Sensor Systems, pp.126–137, ACM, New York, NY, USA, 2003.
[16] W. Zhang, S.K. Das, and Y. Liu, "Security in wireless sensor networks: A survey," in Security in Sensor Networks, ed. Y. Xiao, pp.237–272, Auerbach Publications, 2007.
[17] G.J. Simmons, "Key distribution via shared secret schemes," in Contemporary Cryptology: The Science of Information Integrity, ed. G.J. Simmons, pp.441–497, IEEE Press, 1992.
[18] A. Shamir, "How to share a secret," Commun. ACM, vol.22, no.11, pp.612–613, 1979.
[19] G.R. Blakley, "Safeguarding cryptographic keys," American Federation of Information Processing Societies National Computer Conference, pp.313–317, Arlington, Virginia, USA, Sept. 1979.
[20] E.D. Karnin, J.W. Greene, and M.E. Hellman, "On secret sharing systems," IEEE Trans. Inf. Theory, vol.IT-29, no.1, pp.35–41, Jan. 1983.
[21] ETSI, "Radio broadcasting systems; DAta Radio Channel (DARC): System for wireless infotainment forwarding and teledistribution," Oct. 2002. EN300 751.

Eitaro Kohno received his B.E. degree from Hiroshima Institute of Technology, Japan, in 1990. He received his M.Sc. degree from Tokyo Institute of Technology, Japan, in 1992 and belonged to the doctoral course between 1992 and 1996. From 1996 to 2007, he was a Research Associate of Information Processing Center, Hiroshima City University, Japan. He is currently a Research Associate in the Graduate School of Information Sciences, Hiroshima City University, Japan. Also, he is currently working toward the Ph.D. degree at Tokyo Metropolitan University. His current research interests include network software and wireless sensor networks. He is a member of IEEE (U.S.A.) and IPSJ (Japan).

Tomoyuki Ohta received his B.E., M.Sc., and Ph.D. degrees from Hiroshima City University, Japan, in 1998, 2000, and 2006, respectively. He is currently an Associate Professor in the Graduate School of Information Sciences, Hiroshima City University. His current research interests include mobile communication systems. He is a member of IEEE (U.S.A) and ACM (U.S.A).

Yoshiaki Kakuda received his B.E., M.Sc., and Ph.D. degrees from Hiroshima University, Japan, in 1978, 1980 and 1983, respectively. From 1983 to 1991, he was with Research and Development Laboratories, Kokusai Denshin Denwa Co., Ltd. (KDD). He joined Osaka University from 1991 to 1998 as an Associate Professor. He is currently a Professor in the Graduate School of Information Sciences, Hiroshima City University. His current research interests include network software engineering, assurance networks and mobile ad hoc networks. He is a member of IEEE (U.S.A.) and IPSJ (Japan). He received the Telecom System Technology Award from Telecommunications Advanced Foundation in 1992.

Masaki Aida received his B.S. and M.S. degrees in Theoretical Physics from St. Paul's University, Tokyo, Japan, in 1987 and 1989, respectively; and received the Ph.D. in Telecommunications Engneering from the University of Tokyo, Japan, in 1999. In 1989, he joined NTT Laboratories. In April 2005 to March 2007, he was an Associate Professor of the Faculty of System Design, Tokyo Metropolitan University. He has been a Professor of the Graduate School of System Design, Tokyo Metropolitan University since April 2007. His current interests include traffic issues in computer communication networks. He is a member of the IEEE and the Operations Research Society of Japan.