Resiliency of Secret Sharing Schemes against Node Capture Attacks for Wireless Sensor Networks

Eitaro KOHNO, Tomoyuki OHTA, and Yoshiaki KAKUDA

Graduate School of Information Sciences,
Hiroshima City University
3-4-1 Ozuka Higashi, Asaminami-ku,
Hiroshima, Japan, 731-3194
{kouno,ohta,kakuda}@hiroshima-cu.ac.jp

Abstract

In wireless sensor networks, the data is transmitted by radio waves. Sensor nodes, which are used in wireless sensor networks, have limited computational power and memory size. The data is vulnerable to attack due to the nature of the systems so maintaining confidentiality is an important problem. However, it is difficult to apply general methods of security to wireless sensor networks. Most methods are based on common (secret) key cryptosystems or public key cryptosystems. However, these methods do not protect against node capture attacks.

In this paper, we propose a new distribution method resilient against node capture attacks using Secret Sharing Scheme. In addition, we will confirm the ability of our method to improve resiliency against node capture attacks, comparing it with TinySec, which is the major security architecture of wireless sensor networks.

1 Introduction

Highly confidential information relating to fields such as crime prevention, healthcare, disaster management, 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 insertion of packets, and data compromising. Wireless data transfer uses WSNs, and the sensor node of WSNs has strong limitations on the computational resources and the amount of memory. Another problem faced by the nodes is the energy constraint, as the batteries can run out easily. Some countermeasures against this have been reported. Among them there 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 nodes exhaust the battery, the sink node can receive data by the voting technique.

TinySec [2] is one of the most popular security architectures to keep 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, the idea of using a key creates a new problem, though, all common key and public key cryptosystems have a secret key. The protection of the key is another important problem of the security of WSNs. One of the popular key protection methods was first proposed by Eschenauser and Gligor [6]. 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 to share a key. Their method is known as the random key pre-distribution scheme (RKP). RKP has been improved by their successors [7] [8]. Most of these methods mentioned above become invalid when the keys are compromised. To deal with this drawback, a method of scrambling codes and keys [10] was proposed using code obfuscation techniques [9]. This method required more memory and execution time than TinySec. Furthermore, the sharing key system generates overhead in the time it take to share new keys and control messages. When WSNs start sharing a new key, the control packets are exchanged several times. These become the obstacles of rapid key sharing and the re-key phase that must be overcome.

In this paper, we propose a new distribution method resilient against the node capture attack using Secret Sharing Scheme [11][12][13]. In addition, we confirm the ability of our method to improve resiliency against node capture attack, comparing it with TinySec. We compare the agreement time of our scheme to existing methods.

The rest of the paper is organized as follows: in Section 2, we introduce node capture attacks and some problems with wireless sensor networks. In Section 3, we highlight the proposed scheme, respectively. In Section 4, we discuss the performance of our scheme against node capture attacks. We conclude the proposed method in Section 5.

2 Fault detection and Security on WSNs

2.1 Mica Mote

A Mote [3] 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 devices. 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 other than the communication device.

Table 1: Platforms of Mote and their hardware specifications

Platform	MICAz	MICA2	MICA
CPU	ATmega128		ATmega103
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

The Mote has a special operating system for embedded devices, called TinyOS. The developing environment and tools are distributed as open source software. In addition, we use the simulator TOSSIM [4] to check the behavior on a real machine.

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 hostile environment for a long time to take a measurement. As a result, WSNs may be affected by many kinds of attacks. This section describes the attacks which are targeted in this paper for WSNs.

Zhang [5] et al. list three kinds of attacks to wireless networks: eavesdropping, compromising, and node insertion. Among these, the eavesdropping attack can be prevented by various cryptographic methods using a common (secret) key. TinySec is one of the popular architectures of these methods. However, systems such as TinySec are comparatively weak against node compromise. Figure 1 shows the key being compromised by a node capture. In this paper, a node capture attack is defined

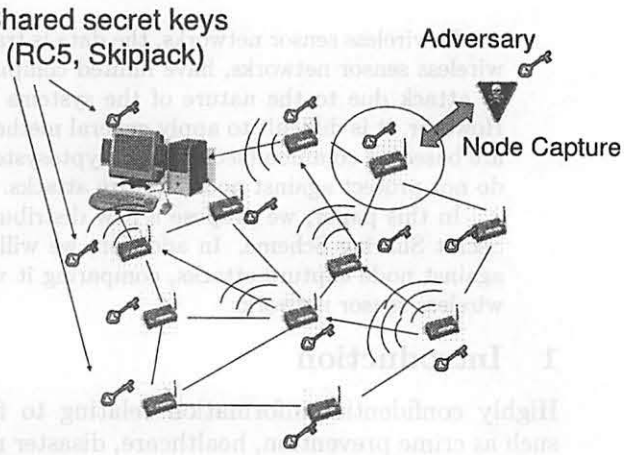

Figure 1: Compromising a common (secret) key from node capture attacks in TinySec

as keys/data inside a node being compromised or stolen. Once the keys/data are compromised, they become invalid and the system is corrupted.

3 Proposed method

3.1 Assumptions

In this section, 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 $p(x)$.

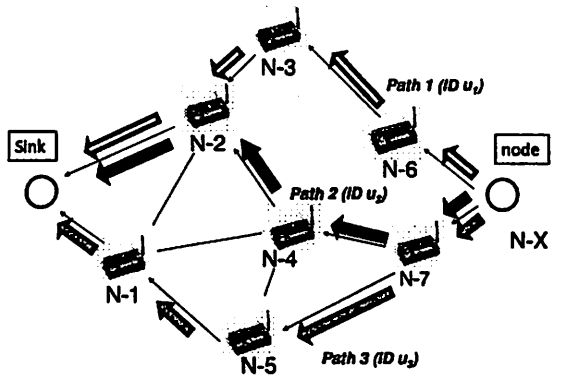

Figure 2: Assumptions of the proposed method

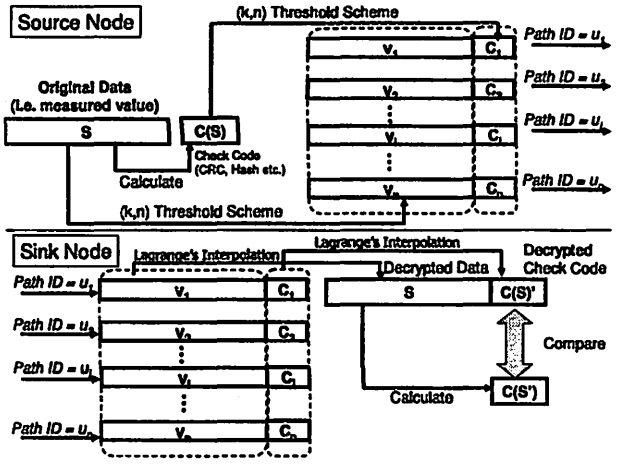

Figure 3: Conceptual diagram of our proposed method

Figure 2 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 routing protocol for the method of finding multiple paths is outside the scope of this paper, the sequence numbers of the route request packets and route reply packets can be utilized for ID.

3.2 Overview and key idea of proposed method

The conceptual diagram of our proposed method is illustrated in Figure 3. Our basic idea is to utilize the dispersal of the information to protect the original data S by the threshold scheme [11][12] on a source node. At the same time, the source node calculates the check code (e.g. check sum, hash without key, and so on.) $C(S)$ before dispersing the original data S. Afterward, the source node calculates each share by the threshold scheme using $f(x) = a_0 + \sum_{j=1}^{k-1} a_j x^j$ over $GF(2^m)$ from the original data, S, and the check code value, $C(S)$, individually. The random integers a_i $(i = 1, 2, \cdots, k-1)$ are generated by each source node.

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 Lagrange's interpolation method[11] [12]. At this time, the source node does not seek an agreement on the threshold value with the sink node. This means the source node can change the threshold value at anytime without having to consult with the sink node.

In a WSN such as in Figure 2, 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 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 the result, WSNs are classified into 4 cases: (a) without any event; (b) with one or more node fault(s); (c) with one or more node capture(s); (d) without any event but re-routing.

If the sink node uses our method for decryption, 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) if the data is decrypted correctly but the threshold number does not match.

If captured nodes fabricate the path ID, the source node creates shares utilizing it. Because of the inconsistency between the fabricated and correct path ID, the sink node can not calculate correct decrypted data.

3.3 Algorithm

We describe the algorithms and the procedures of the source/sink node using Figure 3.

procedure of source node First of all, each source node calculates the CRC (Cyclic Redundancy Code) value, $C(S)$, from the own

original data S. Each source node calculates shares, v_i and c_i, as follows:

$$v_i = S + \sum_{j=1}^{k-1} a_j u_i^j, \quad (1)$$

$$c_i = C(S) + \sum_{j=1}^{k-1} a_j u_i^j, \quad (2)$$

where a_j $(j = 1, 2, \cdots, k-1)$ are random numbers and u_i is path ID. Equations (1) and (2) are calculated over $GF(2^m)$. We employ a (k, n) threshold scheme to produce n shares, where $n \geq k$.

procedure of sink node The sink node uses Lagrange's interpolation method to calculate the decrypted data S' and its CRC value $C'(S)$ using shares from the source node. The accuracy of the decrypted data, S', is checked by comparison between CRC value of decrypted data, $C(S')$, and decrypted CRC value from shares, $C'(S)$.

3.4 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 $p(x) = x^8+x^7+x^2+x+1$ for the threshold scheme, and the CRC-16 function from the ITU-T CRC standard [14] with $g_{16}(x) = x^{16}+x^{12}+x^5+1$. In Table 2, "TinyOS" does not include security support, "TinySec" shows TinySec with Skipjack, and "Proposed" shows our proposed method. Table 2 shows, for each of the three methods, memory size is in acceptable range of RAM and ROM. And the number within the table in parentheses shows the ratio of the values to its "TinyOS." The ROM size of our method is smaller than that of "TinySec," because the calculation of threshold scheme are more light weight than the key based cryptosystems. However, the RAM size of our method is larger than that of "TinySec." It is large due in part to buffer size needed to store each share until finishing the SSS calculation.

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)	12.0 (1.46)
	RAM (Byte)	336	616 (1.83)	933 (2.78)
MICA2	ROM (kByte)	10.5	19.4 (1.85)	13.9 (1.32)
	RAM (Byte)	447	706 (1.58)	1044 (2.34)
MICAz	ROM (kByte)	9.87	- (-)	13.5 (1.37)
	RAM (Byte)	392	- (-)	989 (2.52)

4 Simulation results and discussion

4.1 Experiment for evaluation

We define resiliency against single node capture as the security of WSNs. We determine the reliability of WSNs on the expected number of compromised nodes, which are 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 use the topology illustrated in Figure 4. 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 multipath from the source node to sink was determined automatically by packet loss rate. When the packet

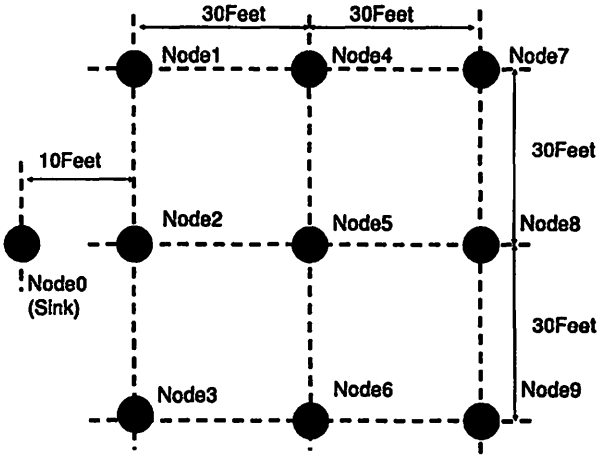

Figure 4: The topology of the experiment

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 $k = n$.

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 3 shows the results of the experiment. All data in Table 3 are the average of 10 trials. The bracketed numbers in Table 3 indicates an increasing ratio based on each value of its "TinyOS." As seen in Table 3 the proposed improvement in the ratio of execution time and the number of packets. Meanwhile, we have eliminated the captured of the sink node by third party. Next, we consider the relation between the shortest number of hops and fraction of data compromised. We determine the fraction of data compromised versus the shortest hops on the probability of stolen data of a node with the shortest hops, when a single node is captured. Figure 5 shows the fraction of data compromised versus the shortest hops. Figure 5 indicates that our proposed method becomes more effective as the number of hops increases.

On the shared secret or public key based cryptosystems, the system needs to re-key when it detects compromising keys and so on. When the system changes the keys of two nodes, the nodes must exchange the information to create new keys a number of times. As a result, the key based systems affect the length of the hops. However, our system can change the threshold number and the polynomial $f(x)$ without any agreement. This is the significant advantage over the key based systems.

4.2 Overhead

Now we will look at the overhead of nodes that have 1 hop to the sink node. We modified the proposed method so as not to use it on a node that has 1 hop. Table 4 shows the results of our observed data. In Table 4, "Modified" means the modified method mentioned above; "Proposed" means 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 of the resiliency (of single node capture attacks).

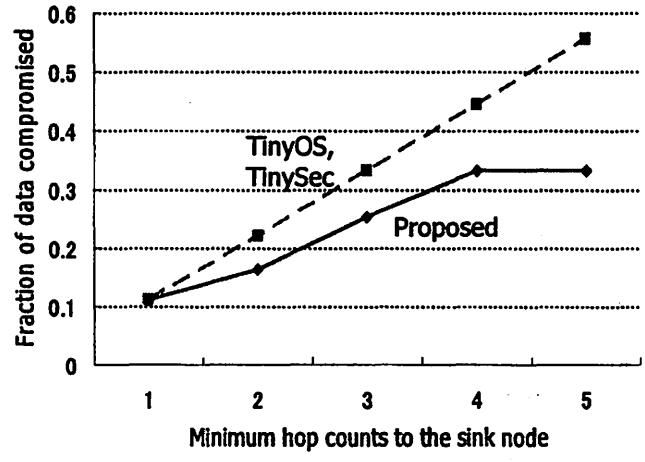

Figure 5: Fraction of data compromised and shortest hops from nodes to sink

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)

5 Conclusion

In many cases, the sensor nodes are exposed to the environment for a long time. Therefore node

capture attacks associated with stealing nodes are important to take into account when considering network security.

In this paper, we proposed applying the Secret Sharing Scheme as a new method resilient to node capture attacks. 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 against node capture attacks than the existing TinySec system is. Additionally, we found our method tends to be more effective as the number of hops to sink node increases. On the other hand, we were able to observe an increased overhead on shorter hop nodes. We have shown a countermeasure capable of reducing excess dispersals without degrading the resilience of node capture attacks.

Acknowledgements

This work was supported in part by the Ministry of Internal Affairs and Communications of Japan under Grant-in-Aid for Strategic Information and Communications R&D Promotion Programme (SCOPE), the Ministry of Education, Science, Sports and Culture of Japan under Grant-in-Aid for Scientific Research (C) (No.18500062) and Hiroshima City University under their research grants.

References

[1] I. Stojmenović (ed.), "Handbook of Sensor Networks: Algorithms and Architectures," Wiley, 2005.

[2] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks," Proc. 2nd Int. Conf. on Embedded Networked Sensor Systems (SenSys'04), pp.162–175, Baltimore, Maryland, USA, November 2004.

[3] 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.

[4] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications," Proc. 1st Int. Conf. on Embedded Networked Sensor Systems (SenSys'03), pp.126–137, Los Angeles, California, USA, November 2003.

[5] W. Zhang, S. K. Das, and Y. Liu, "Security in Wireless Sensor Networks: A Survey," In Security in Sensor Networks, Y. Xiao ed., Auerbach Publications, pp.237-272, 2007.

[6] L. Ecshenauer, and V. D. Gligor, "A Key-management Scheme for Distributed Sensor Networks," Proc. 9th ACM Conf. Computer and Communication Security (CCS '02), pp.41–47, Washington, DC, USA, November 2002.

[7] A. C. Chan, "Probabilistic Distributed Key Pre-distribution for Mobile Ad hoc Networks," Proc. IEEE Int. Conf. on Commun. (ICC 2004), vol.6, pp.3743–3747, Paris, France, June 2004.

[8] H. Chan, A. Perrig, D. Song, "Random Key Predistribution Schemes for Sensor Networks," Proc. IEEE Symposium on Security and Privacy, pp.197–213, Oakland, California, USA, May 2003.

[9] C. Linn, and S. Debray, "Obfuscation of Executable Code to Improve Resistance to Static Disassembly," Proc. 10th ACM Conf. on Computer and Commun. Security (CCS '03), pp.290–299, New York, NY, USA, 2003.

[10] A. Alarifi, and W. Du, "Diversify Sensor Nodes to Improve Resilience Against Node Compromise," Proc. 4th ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'06), pp.101–112, Alexandria, Virginia, USA, October 2006.

[11] G. J. Simmons, "Key Distribution via Shared Secret Schemes," Contemporary Cryptology: The Science of Information Integrity, pp.441–497, IEEE Press, 1992.

[12] A. Shamir, "How to Share a Secret," Commun. ACM, vol.22, no.11, pp.612–613, 1979.

[13] G. R. Blakley, "Safeguarding Cryptographic Keys," Proc. American Federation of Information Processing Societies National Computer Conf., vol.48, pp.313–317, Arlington, Virginia, USA, September 1979.

[14] ETSI, "Radio broadcasting systems; DAta Radio Channel (DARC): System for wireless infotainment forwarding and teledistribution," ETSI, EN300 751, V1.2.1, October 2002.