리플레이 공격 방어를 위한 무선 센서 네트워크 보안 프로토콜
Resilient Security Protocol for Combating Replay Attacks in Wireless Sensor Networks

장 적*, 허 웅*, 유강수**, 최재호*
전북대학교 전자공학부 영상정보신기술연구센터*,
전주대학교 교양학부**

Di Zhang(zhangdi_cn@jbnu.ac.kr)*, Ung Heo(heoprinc@jbnu.ac.kr)*,
Kangsoo You(gsyou@jj.ac.kr)**, Jaeho Choi(wave@jbnu.ac.kr)*

요약

자원이 제한되어 있는 센서 노드에 보안 프로토콜 기능을 부여할 때에는 항상 어려움이 동반된다. 다양한 인증 기법들 중에서 NEKAP은 지역적 특성을 활용하는 비교적 효과적인 프로토콜이다. 반면, 이 기법은 룹홀 형성이 가능하여 리플레이 공격에 취약한 점이 있다. 본 논문에는 NEKAP의 약점을 보완하여 무선 센서 네트워크가 리플레이 공격을 효과적으로 방어할 수 있도록 향상된 인증 기반 보안 프로토콜을 제안한다. 제안한 기법에서는 네 개의 보안 키를 각 노드에 부여하여 지역과 계층적 요구에 적합한 보안을 제공한다. 제안한 기법의 성능은 보안 인증, 공격 노드의 탐지, 리프레이 공격 방어 등의 측면에서 분석적 기법을 사용하여 평가하였으며 기존의 기법과 비교하였을 때 제안한 기법이 더 좋은 결과를 나타내었다.

■ 중심어 : | 무선 센서 네트워크 | 보안인증 | 리플레이 공격 | 보안 프로토콜 |

Abstract

Due to the resource limitations of sensor nodes, providing a security protocol is a particular challenge in sensor networks. One popular method is the neighborhood-based key agreement protocol (NEKAP). NEKAP is an efficient and lightweight protocol, but it includes loopholes through which adversaries may launch replay attacks by successfully masquerading as legitimate nodes. In this paper, we present a modified security protocol for wireless sensor networks. We provide four types of keys for each node, which adapt to different security requirements; and an improvement is made to alleviate the replay attack. According to our qualitative performance analyses, the proposed security protocol provides effectiveness in terms of authentication security, attacking node detection, and replay attack resilience when compared to the conventional method.

■ keyword : | Wireless Sensor Networks | Authentication | Replay Attack | Security Protocol |

I. INTRODUCTION

Wireless sensor networks (WSNs) are distributed systems consisting of a large number of sensor nodes with a base station as a controller that serves as the interface between the sensor network and the outside

접수번호 : #100504-003
접수일자 : 2010년 05월 04일
심사완료일 : 2010년 06월 16일
교신저자 : 최재호, e-mail : wave@jbnu.ac.kr

network. WSNs may be deployed in unattended and adversarial environments such as battlefields. Compared to conventional networks, they are more vulnerable to physical destruction and man-made threats. Therefore, providing security is a particular challenge in sensor networks due to the resource limitations of sensor nodes, wireless communications, and other related concerns. As a specific example, it is impractical to use asymmetric crypto-systems in sensor networks in which each node has low operational capability and insufficient memory (e.g. Crossbow's MICA2/MPR400CB sensor node [1]). Thus, the key management protocols for sensor networks are based upon symmetric key algorithms, and the design of the security protocols for WSNs should be as lightweight as possible.

NEKAP [2] is a link layer key agreement protocol for sensor networks that establishes two kinds of keys: pairwise keys, for link layer pairwise communications, and cluster keys, for link layer broadcast communication. In NEKAP, the node keys are generated from the master keys of the neighbor nodes, making the discovery of these keys more difficult for enemies. To establish all of the keys, each node broadcasts only three messages, so the protocol is very energy-efficient. The main contribution of NEKAP is the implementation of a key agreement in which each key is valid only in its neighborhood, and therefore the impact of a compromised node key can be restricted to that node's neighborhood. Thus, it is impossible for an adversary to carry out a wide-scale attack by capturing only a few nodes. Moreover, the energy cost of this solution is lower than that of previous solutions.

NEKAP has many advantages for WSNs because it is intruder resilient and energy efficient. Unfortunately, NEKAP is vulnerable to replay attacks [3] because of the key establishment process, which includes only three broadcast messages. A malicious node may transmit an old message that was originally broadcasted from a legal node to its neighbor nodes, and the message cannot be authenticated because these two nodes cannot communicate directly (see Section Ⅱ). Therefore, a malicious node may gain legal status by cheating the chosen legal nodes by transmitting the old message, and then an adversary may launch other attacks, such as DOS [4] attacks, black-hole attacks, or masquerade attacks. In addition, NEKAP suffers from node tampering during network initialization. The problem stems from an irrational assumption made on the relationship between the secure key establishment time and the node tampering time. It will be discussed in detail in Section Ⅱ.

Therefore, this work is motivated by solving the drawbacks in both LEAP and NEKAP protocols without carrying more resource consumption. In this paper, we present a modified NEKAP that can prevent replay attacks, and we present a new modified security protocol for wireless sensor networks. The focus of the paper is to dismantle the unreasonable time assumption made in NEKAP and making the authentication and security protocol of us be much general and resilient.

The rest of the paper is organized as follows. In Section Ⅱ, we review related studies that have previously presented security protocols for sensor networks, provide an overview of NEKAP, and we describe loopholes in NEKAP that may be exploited by adversaries to launch replay attacks. In Section Ⅲ, we discuss our system and assumptions, and present the details of our modified security protocol. We present a security and performance analysis in Section Ⅳ, and provide our conclusions in Section Ⅴ.

II. RELATED WORKS

Link layer key agreements between neighboring nodes are fundamental for securing sensor networks deployed in unattended and hostile environments [5]. There are several relevant approaches presented in the literature [6-8]. Link layer key agreements allow two nodes to communicate directly via a shared pairwise key.

The localized encryption and authentication protocol (LEAP) [9] was first proposed by Zhu, et al., as a key management protocol for sensor networks designed to support in-network processing. LEAP solves the problem of key distribution and restricts the impact of a compromised node on the network. LEAP establishes four types of keys, for each node and communication type: 1) the individual node key, which is shared between each node and the base station and is used to communicate with the base station, is pre-loaded before its deployment; 2) the pairwise key, which is shared between a node and each one of its neighbors, is used in pairwise communication among them; 3) the cluster key, which is shared between a node and all of its neighbors, is used in local broadcast communication; and 4) the group key, which is shared by all nodes, is used in broadcast multi-hop from the base station. In sensor networks, the use of a pre-deployed key is the most practical approach for bootstrapping secret keys in sensor nodes. In LEAP, the nodes were loaded before they were deployed in the sensor field. Pairwise keys could be generated between two nodes based on this pre-deployed key information. The problem with LEAP is the excessive number of messages that must be exchanged during the establishment of the keys; the communication cost is very high.

Oliveira, et al., presented SPINS [10], a security protocol for WSNs, and proposed two building security blocks optimized for sensor networks: SNEP and μTESLA. SNEP provides end-to-end data confidentiality, two part data authentication, and data freshness between the base station and each node; μ TESLA is a protocol that provides multihop broadcasting from the base station.

Since NEKAP is a peer-to-peer approach, it can be used in combination with the SNEP or μTESLA protocols to increase security for sensor networks.

1. Overview of NEKAP

NEKAP is a link layer key management protocol that establishes two kinds of keys: pairwise keys, for link layer pairwise communication; and cluster keys, for link layer broadcast communication. It is similar to LEAP, however, NEKAP is more resilient to node tampering and is even more energy-efficient.

In NEKAP, each node is pre-loaded with a master key, and broadcasts to its neighbors using this key encrypted with a global shared key. The node keys are generated from the master keys of neighbor nodes, making the discovery of these keys more difficult for an adversary. To establish all of the keys, each node broadcasts only three messages, so the protocol is very energy-efficient.

Since the key is valid only within its neighborhood, and since the impact of a compromised node key can be restricted to the node's neighborhood, NEKAP is also intruder resilient.

2. Loopholes of NEKAP

In NEKAP, the process of key establishment consists of only three broadcast messages, which are broadcast from each node to its neighbor nodes. NEKAP can provide data confidentiality, but it cannot provide broadcast authentication during the key establishment phase. Thus, the nodes are vulnerable to replay attacks.

In replay attacks, malicious nodes are deployed in a sensor network during the initialization phase. If the malicious node retransmits legitimate old messages previously broadcast from a legal node to another that cannot communicate directly, the malicious node can pass itself off as a legal node in the network, as shown in [Figure 1].

In [Figure 1], the malicious node retransmits node A's broadcast messages to node B, so that node B will then regard the malicious node as node A. Similarly, the malicious node can also act as a neighbor node B to node A if it retransmits node B's broadcast messages to node A. Actually, however, nodes A and B cannot communicate with each other directly, and the malicious node acts as an intermediate node between nodes A and B in the network. The malicious node cannot threaten the security of its region when it is between two nodes that can communicate directly, however. [Figure 1] shows an example of an attack by one node, and [Figure 2] shows an example of an attack by two nodes.

Combining these two conditions, random diffusion with several malicious nodes will confuse the framework of the network (as shown in [Figure 3]). The adversary can then execute a DOS attack or a black-hole attack after the routing is established.

Another problem comes from the time assumption made in NEKAP (also in LEAP). It assumes that there exists a lower bound on the time interval Tmin, which is a necessary time for an adversary to compromise a sensor node, and that the time Test for a newly deployed sensor node to discover its immediate neighbors is smaller than T_{min}. Even though it can be practically a reasonable assumption that $T_{min} > T_{est}$, however, if an adversary can compromise a sensor node within the time interval T_{est}, it can discover all of the information in the node, and can then decrypt all of the broadcasting information using the taken global key.

In the next section, we present a better protocol that can alleviate the time assumption and improve resilience against replay attack while keep the resource consumption at the comparable level as in conventional method.

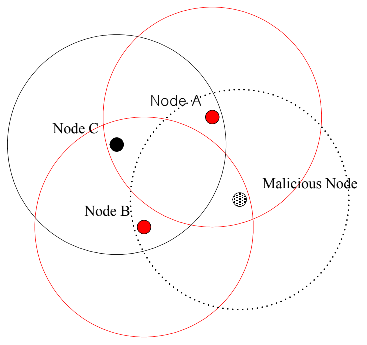

Fig. 1. Replay attack by one node

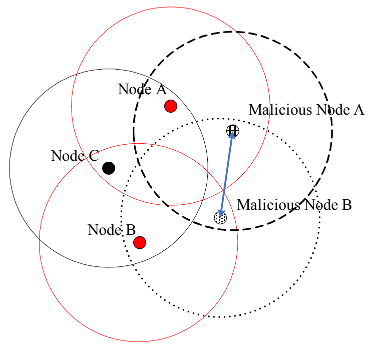

Fig. 2. Replay attack by two nodes

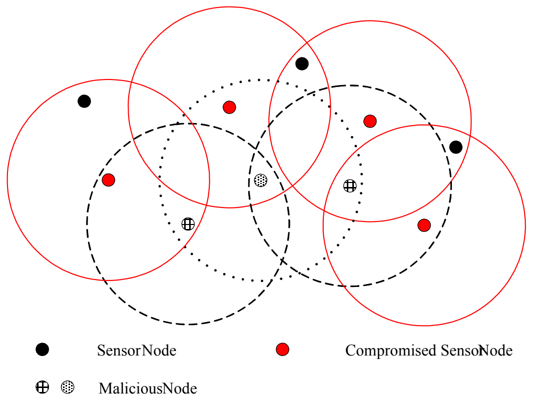

Fig. 3. Replay attack by more than two nodes

III. PROPOSED METHOD

1. System and Assumptions

We assume that a typical sensor network forms around one or more base stations acting as the controller (or key server) with sufficient power, memory, and computational capabilities to serve as the interface between the sensor network and the outside network. The sensor nodes establish a routing forest, with a base station at the root of every tree. However, we assume that the base station will not be compromised. In such a system, node deployment is random, the neighborhood of any node is not known in advance, the wireless communication is not secure, and the system is subject to eavesdropping, package insertion, and replay of older messages. The nodes are vulnerable to tampering. We assume that if a node has been compromised, the enemy has access to all of the information handled by that node.

2. Notation

The following symbols are used in the text:

- ID : Node identifier, MAC address;
- f : Pseudo-random function;
- K_G : Global key shared in each node;
- K_M : Master key, known only by the BS;
- K_A : Individual key of node A;
- K_A' : Cluster key of node A;
- K_{A_n}: The nth key of node A's one-way key chain for local broadcast authentication;
- K_A^{Id} : Identification key of node A;
- K_{IM} : Identification master key, known only by the BS;
- K_{AB} : Pairwise key shared between nodes A and B;
- $K_{\in}$: The insertion key used for new node insertion in the insertion phase;
- $BS \Rightarrow *$: Broadcast message sent by BS;
- $\{M\}_K$: The encryption of message M with encryption key K;
- $\{M\}_{(K,IV)}$: Denotes the encryption of message M, with key K and the initialization vector IV which is used in encryption modes;
- $MAC\{M\}_K$: Denotes the computation of the message authentication code (MAC) of message M, with MAC key K.

3. Protocol Description

As in LEAP and NEKAP, the design of our protocol supports multiple keying mechanisms, following the observation that different types of messages exchanged between sensor nodes have different security requirements, and that a single keying mechanism is not suitable for meeting all of these different security requirements. Specifically, we support the establishment of four types of keys for each sensor node: an individual key shared with the base station, a pairwise key shared with another sensor node, a cluster key shared with multiple neighboring nodes, and a global key shared by all of the nodes in the network.

Our protocol also includes an efficient protocol for local broadcast authentication based on the use of one-way key chains.

In order to prevent replay attacks, our protocol provides a malicious node detection phase to detect and remove any malicious nodes that may exist in the network.

In addition, we provide a new bootstrapping method in our protocol that solves the security threat of the initialization phase (detailed in Section Ⅳ).

Our procedure is described as follows:

1) *The Initialization Phase*

Step 1: Each node is pre-loaded with a unique number as its node identifier(ID).

Step 2: The controller creates a master Key (K_M) and an identification master key(K_{IM}) for all nodes that is known only by the base station(BS).

Step 3: Compute and install an identification key (K_A^{Id}) for each node A:

$$K_M^A = f(K_{IM}, A) \tag{1}$$

Step 4: Each node A pre-loads its individual key (K_A), cluster key($K_A^{'}$), and global key(K_G):

$$K_A = f(K_M, A) \tag{2}$$
$$K_A^{'} = f(K_M, A)$$

2) *The Broadcast Phase*

Step 1: When the broadcast phase starts each node broadcasts a message to its neighbor nodes:

$$A \Rightarrow *: \left\{K_{A_1}, A, K_A^{'}\right\}_{(K_G, K_A^{Id})'} A \tag{3}$$

Step 2: There is a short waiting phase for all nodes to complete broadcasting of messages.

Step 3: The base station (BS) broadcasts and reveals the identification master key(K_{IM}) to all nodes.

$$BS \Rightarrow *: K_{IM} \tag{4}$$

The neighbor nodes can compute the identification key (K_A^{Id}) of node A, and they can then decrypt the packet to get the cluster key of node A, the first key of node A's one-way key chain for local broadcast authentication and verify the identification of the packet. At last, the identification key (K_A^{Id}) and identification master key (K_{IM}) are erased.

Step 4: When the node has finished the above process, it will broadcast its neighbor nodes list to its neighbor nodes:

$$A \Rightarrow *: \{ID_i | i \in N_A\}_{K_A^{'}}, K_{A_2}, \tag{5}$$
$$MAC(\{ID_i | i \in N_A\}_{K_A^{'}}, K_{A_2})_{K_A^{'}}$$

Node A's neighbor nodes B can receive the list from node A, and the pairwise key(K_{AB}) between nodes A and B will be:

$$K_{AB} = f(K_A^{'}, K_B^{'}, ID_i | i \in N_A \cap N_B) \tag{6}$$

The pairwise key between nodes A and B is computed using their cluster keys and the identifiers of their common neighbors. This makes it more difficult for adversaries to compromise the network. For example, in [Figure 1], the common neighbor of nodes A and B is node C.

3) *Malicious Node Detection and Diagnosis Phase*

In a replay attack, a malicious node stores a received message and attempts to send it at a later time. When the nodes receive the message, they believe that it is an original message, even though it is not. That causes the nodes to calculate incorrect distance and signal strength since the node sending the original message is not where they think it is.

Most proposals for preventing a replay attack rely on a timestamp or sequence number. The timestamp method must be supported by a synchronization mechanism, which is a complex computation and therefore consumes a great deal of energy. The method based on the sequence number is not applicable to our sensor network since the compromised nodes cannot communicate directly.

In our protocol, we implement a malicious node detecting and diagnosing mechanism based on the acknowledgment message (ACK) to solve the above problems.

Step 1: When node A receives a message from node B, an ACK is generated and sent to node B. This ACK message must be encrypted by the pairwise key(K_{AB}) to avoid fabrication by an adversary.

$$A \Rightarrow B : \{ACK\}_{K_{AB}}, MAC(\{ACK\}_{K_{AB}})_{K_{AB}} \qquad (7)$$

The message is then saved in a temporary buffer until the ACK comes back.

Step 2: Node B resets timer when it receives the ACK, and then it decrypts the MAC message to verify this ACK. If the ACK is authentic, node B will add a timestamp TB before this message is sent back to node A.

$$B \Rightarrow A : \{ACK, T_B\}_{K_{AB}}, MAC(\{ACK, T_B\}_{K_{AB}})_{K_{AB}} \qquad (8)$$

If the ACK is received in a certain amount of time, then the node is an honest node, but if the message is not received in that amount of time, then it is a dishonest node, and the message may have been transmitted by a malicious node. Thus, the node will erase all related information, such as the pairwise key. (The procedure is shown in [Figure 4]).

4) *New Node Insertion Phase*

The new node insertion phase is the same as that of NEKAP, so we have omitted the details of this phase.

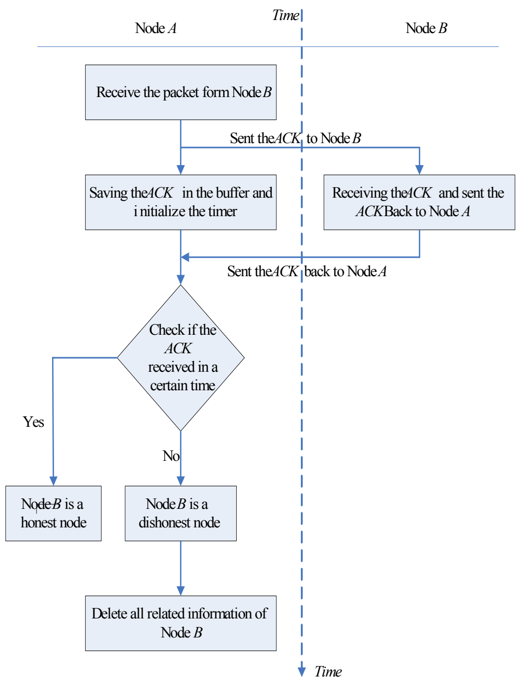
Node A
Time
Node B
Receive the packet form Node B
Sent the ACK to Node B
Saving the ACK in the buffer and initialize the timer
Receiving the ACK and sent the ACK Back to Node A
Sent the ACK back to Node A
Check if the ACK received in a certain time
Yes
No
Node B is a honest node
Node B is a dishonest node
Delete all related information of Node B
Time

Fig. 4. The procedure for malicious node detection and diagnosis

IV. SECURITY AND PERFORMANCE ANALYSIS

In this section, we discuss some issues and problems regarding our modified protocol.

1. Security Analysis:

1.1 Key discovery by the adversary

The keys provided in our protocol can be used to authenticate all link layer messages. By discarding non-authenticated messages, the nodes provide access control to the network communication. Access control prevents external nodes from successfully implementing several kinds of attacks such as the insertion of false data, data modification, spoofing, and attacks of denial of service in routing including black hole, selective forwarding, and wormhole attacks. Therefore, an enemy can use internal attacks only, and to do this it needs to have information about the keys.

The adversary can discover the keys by eavesdropping and cryptanalysis or by tampering with a node, which are expensive processes. These attacks can only reveal a limited number of keys during network operation.

Due to several similarities, LEAP[9] and NEKAP[2] were chosen for comparison with our modified protocol (as shown in [Table 1]).

[Table 1] presents a comparison between the number of keys that can be discovered using several kinds of attacks in LEAP, NEKAP, and our modified method. We can see that both LEAP and NEKAP are vulnerable during the network initialization phase, since their bootstrapping methods for key establishment are almost the same. For that reason, we have developed a new bootstrapping method.

1.2 About the new bootstrapping method

LEAP and NEKAP are based on an important shared assumption. They assume that there exists a lower bound on the time interval $T_{\min}$ that is a necessary time for an adversary to compromise a sensor node, and that the time T_{est} for a newly deployed sensor node to discover its immediate neighbors is smaller than $T_{\min}$. In reality, it seems a reasonable assumption that $T_{\min} > T_{est}$, but if an adversary can compromise a sensor node within the time interval T_{est}, they can discover all of the information in the node, and can then decrypt all of the broadcasting information using the global key.

Table 1. Key Discovery Comparison

During operation or initialization phases			
Attack	LEAP	NEKAP	Modified Method
1. Node tampering during network operation	Only node keys	Only node keys	Only node keys
2. Node tampering during network initialization	All keys of network	Some keys	No effect
3. Global key discovery during network operation	All keys of network	None	None
4. Global key discovery during network initialization	All keys of network	All keys of network	None

In our protocol, we don't need the time assumption as in NEKAP, because we provide the new bootstrapping method. In LEAP and NEKAP, the message is encrypted by the global key, which should be pre-loaded to all of the nodes at the initialization phase. In chance, the adversary can compromise a node and get the information e.g. master keys or global key, then fabricate some message to intrude into the network. But in our protocol, each node sends "$\{K_A, A, K_A'\}_{(K_{G, K_A^M})}, A$" to its neighbor nodes. Notice that each node does not know the identification key (K_A^M) of any other node since that is constructed from the identification master key (K_{IM}). Even if the adversary compromises a sensor node within the time interval T_{est}, it cannot decrypt any packets. The adversary does not know any node's information except that of the compromised node before the identification master key (K_{IM}) is revealed. In other words, this packet cannot be fabricated and falsified, since nobody can decrypt the message without the identification master key (K_{IM}), and the identifier contained in the packet can authenticate the same one outside.

The difference between NEKAP and our modified protocol is that the nodes in the cluster group can complete keys exchanges safely because these two master keys is only known by the base station, and these keys are release only during a meaningful duration so that the keys are used securely at the initiation time.

The construction of the pairwise key is also based on the neighbor nodes. Therefore, our modified protocol also has the positive characteristics of NEKAP, including the fact that it is intruder resilient.

1.3 About the malicious node detecting and diagnosing mechanism

In order to solve the threat of a replay attack, we present a malicious node detecting and diagnosing mechanism based on the acknowledgment message.

MAC layer timestamping is an effective method for improving the precision of our malicious node detecting and diagnosing mechanism. However, it cannot be used without adaptation. Since an ACK packet is encrypted after it is time stamped, the sending time of the packet will be later than the timestamp. The difference between the two times consists of the encryption time ($t_{encryption}$), the MAC calculation time ($t_{MAC\text{-}calculation}$), and the transmission time of the timestamp signal ($t_{timestamp}$), as shown in [Figure 5].

$$\begin{aligned}\triangle_{error} &= T_y - T_x \\ &= t_{timestamp} + t_{enryption} + t_{MAC-calculation} \\ &= \frac{l_{timestamp}}{s} + l_{packet} \times \alpha + t_{MAC-calculation}\end{aligned} \tag{9}$$

Here, α is the encryption time of a single byte.

In our mechanism, the timestamp is added at the moment t_x before the packet is encrypted, as shown in [Figure 5]. We set the timestamp to the time t_y when the packet is actually sent in the MAC layer.

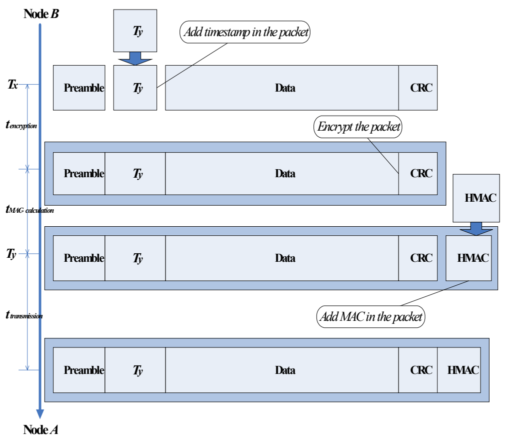

Fig. 5. Error of timestamping in MAC layer when the *ACK* message is sent back to Node *A*

This method takes advantage of the fact that sending ACK through a malicious node would take longer than if it were transmitted directly. If the ACK is received in a certain amount of time, then the node is an honest node, but if the message is not received in that amount of time then it is a dishonest node, and the message may have been transmitted by the malicious node. The amount of time ($T_{threshold}$) can be more precise with the help of $\triangle_{error}$.

$$t_{threshold} = t_{transmisson} + t_y = t_{\overrightarrow{AB}} + t_{\overrightarrow{BA}} + T_x + \triangle_{error} \tag{10}$$

Here, $t_{\overrightarrow{AB}}$ is the duration of the transmission time for the *ACK* message to be sent from node *A* to node *B*; $t_{\overrightarrow{AB}}$ is the duration of the transmission time from node *B* to node *A*.

Therefore, proper detection and diagnosis of malicious nodes will help our sensor network safely build a routing table.

This is a lightweight and effective protocol. The controller can detect and define most malicious nodes, and can then dispose of the malicious nodes.

2. Performance Analysis:

We consider the following performance metrics in our protocol.

2.1 Communication Overhead

Let n be the total number of nodes, let v be the average number of neighbors, and $N(n)$ be the number of all messages transmitted for key establishment.

The cost of node deployment for LEAP in terms of the number of transmitted messages is one message from node (n), a response from each neighbor in the neighbor discovery phase(n,v), and one message to each neighbor for the cluster key announcement (n,v), resulting in N(n,v):

$$\begin{aligned} N(n,v) &= 1\times n+1\times v\times n+1\times v\times n \\ &= (1+2v)n \end{aligned} \quad (11)$$

In NEKAP, the cost of node deployment is one message from node (n), one message to send the master key (n), and one message to complete the neighbor announcement, resulting in $N(n)$:

$$\begin{aligned} N(n) &= 1\times n+1\times n+1\times n \\ &= 3n \end{aligned} \quad (12)$$

In our protocol, the cost of node deployment is only one message to send the master key (n), plus one message to complete the neighbor announcement (n), resulting in $N(n)$:

$$\begin{aligned} N(n) &= 1\times n+1\times n \\ &= 2n \end{aligned} \quad (13)$$

Thus, in our protocol, there are only two broadcast messages that need to be transmitted during the key establishment phase, and the malicious node detecting and diagnosing mechanism is a lightweight protocol, so the communication overhead required is very low.

2.2 Computational Overhead

The main computational overhead for each node is to verify a *MAC* and to establish a pairwise key with every neighbor node. All of these processes are easy to complete, so the computational overhead is also low.

Table 2. Communication Overhead Comparison

Messages transmitted for key establishment	LEAP	NEKAP	Modified Method
N(n,v)	(1+2v)n	3n	2n

V. CONCLUSIONS

We have presented a modified protocol for wireless sensor networks, which not only has the advantages of NEKAP, but also solves some of the security problems of NEKAP. The properties of our protocol are as follows:

- Our protocol supports four types of keys per node. These keys can be used to increase the security of many non-secure protocols.
- We use a new bootstrapping scheme during the key establishment phase to prevent an adversary from compromising a sensor node.
- Our protocol provides lightweight, effective malicious node detection and diagnosis based on the acknowledgment message.
- To generate the keys, our protocol requires only two broadcasted messages from each node, and therefore is energy-efficient and appropriate for use in WSNs.

참 고 문 헌

[1] Http://www.xbow.com/Support/Support_pdf_files/getting_started_guide.pdf

[2] S. De Oliveira, H. C. Wong and J. M. Nogueira, "NEKAP: Intruder Resilient and Energy Efficient Key Establishment in Sensor Networks," Proc. of ICCCN'07, pp.803-808, Honolulu, Hawaii, USA, 2007.

[3] M. Vella and A. Mahdy, "Survey of wireless sensor network security," Proc. of SACNAS '08, pp.128-134, Salt Lake, Utah, USA, 2008.

[4] D. Raymond and S. Midkiff, "Denial-of-Service in Wireless Sensor Networks: Attacks and Defenses," IEEE Pervasive Computing, Vol.7, No.1, pp.74-81, 2008.

[5] Y. Zhou and Y. Fang, "Scalable Link-Layer Key Agreement in Sensor Networks," Proc. of MILCOM '06, pp.1-6, Washington, D.C., USA, 2006.

[6] L. Eschenauer and V. D. Glicor, "A key-management scheme for distributed sensor network," Proc. of 9th ACM conference (CCS '02), pp.41-47, Washington, D.C., USA, 2002.

[7] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," 2003 IEEE Symposium on Security and Privacy, Berkeley, pp.197-215, CA, 2003.

[8] D. Liu, P. Ning and R. Li, "Establishing Pairwise Keys in Distributed Sensor Networks," ACM Transactions on Information and System Security, Vol.8, No.1, pp.41-77, 2005(2).

[9] S. Zhu, S. Setia and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks," 10th ACM Conference (CCS '03), pp.62-72, Washington D.C., USA, 2003(10).

[10] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," in *Wireless Networks,* pp.521-534, Kluwer Academic Publishers, Netherlands, 2002.

저 자 소 개

장 적(Di Zhang) 준회원

- 2008년 7월 : South-Central University for Nationalities(공학사)
- 2010년 7월 : 전북대학교 전자공학과(공학석사)

<관심분야> : cryptography, security protocols

허 웅(Ung Heo) 정회원

- 2002년 2월 : 전북대학교 컴퓨터공학과(공학사)
- 2004년 2월 : 전북대학교 컴퓨터공학과(공학석사)
- 2004년 2월 ~ 현재 : 전북대학교 컴퓨터공학과(박사수료)

<관심분야> : 모바일 통신 프로토콜 성능 모델링

유 강 수(Kangsoo You) 정회원

- 1994년 2월 : 전북대학교 컴퓨터공학과(공학석사)
- 2006년 8월 : 전북대학교 영상공학과(공학박사)
- 2006년 9월 ~ 현재 : 전주대학교 교양학부 교수

<관심분야> : 영상처리, 멀티미디어시스템

최 재 호(Jaeho Choi) 정회원

- 1985년 5월 : NCSU(B.S.E.E)
- 1988년 5월 : NCSU(M.S.E.E)
- 1993년 5월 : NCSU(Ph.D in Computer Eng)
- 1994년 3월 ~ 현재 : 전북대학교 전자공학부 교수

<관심분야> : 센서 네트워크, 애드 혹 네트워크