Attribute-based Group Key Management for Wireless Sensor Networks
A Cross-layer Design Approach for Group Key Management

Jun Noda
Cloud System Research Laboratories
NEC Corporation
1753 Shimonumabe, Nakahara-ku, Kawasaki 211-8666 JAPAN
Email: j-noda@cw.jp.nec.com

Yuichi Kaji
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0101 JAPAN
Email: kaji@is.naist.jp

***Abstract*—In this study, we investigated a group key management scheme that is especially suitable for large-scale wireless sensor networks (WSNs). Practical large-scale WSNs typically contain multiple groups of nodes, and the managing server needs to keep a number of group keys secure against possible attacks. We have developed a flexible and versatile formalization of an attribute-based group structure. The proposed formalization can model practical groups in real applications and enables secure and efficient management of multiple group keys. One of the key advantages of this approach is that a certain cross-layer design mechanism can be implemented in the group key revocation protocol. We show through computer simulation that adding some controlled redundancy in the upper-layer protocol helps reduce the lower-layer traffic in a realistic setting. The results demonstrate that using attribute-based groups is more suitable and practical for WSNs than the conventional group key management mechanisms.**

Keywords*-wireless sensor network; security; group key; cross-layer design.*

I. Introduction

Security is a crucial issue in many of the applications used in wireless sensor networks (WSNs). In this study, we focus on providing security through the management of cryptographic group keys owned by sensor nodes.

In a practical WSN with a large number of sensor nodes, nodes are typically sorted on the basis of their attributes, and the resulting *groups* of nodes are organized in the network. A group can function as a unit for the access control of critical information, so it is convenient if all nodes in a group are provided with an identical *group key* that is used to encrypt or authenticate critical data. The group key must be managed in such a way that it is known to group members only. When a node becomes compromised and is removed from the group, we need to replace the group key to prevent the removed node from accessing critical information. In this procedure, called *key revocation*, the key managing *server* selects a new group key and delivers it to all nodes that remain in the group.

Secure and efficient schemes for group key revocation have long been studied in the research arm of the information security field (see [2][6] for short surveys on the key findings of these studies). Unfortunately, most of these schemes are too complicated for WSNs, excepting a few simple schemes such as the LKH protocol [7]. Efforts have also been made to establish simple, lightweight mechanisms that explicitly focus on WSNs. For example, a simple unicast-based scheme can be constructed over SPINS [5]; LEAP [8] makes use of an internal timer to destroy critical information in each node, and [1] applies the idea of self-healing key distribution to WSNs.

Despite the insight gained from these previous works, we feel that some of the important aspects of group key management in WSNs have not been sufficiently considered thus far. For example, these other studies often assume that there is only one group in the network and therefore focus on the management of just one group key. In our view, this is fatally misleading approach. In a practical WSN, there are many groups in the network, and each group is typically managed by a single server. This means that the server can make use of some of the “safe” group keys in the revocation procedure of the “threatened” group keys. Such an organic use of multiple group keys helps make the key revocation much more efficient than managing multiple group keys separately and independently. Having multiple groups in the network is advantageous in terms of realizing secure and efficient key management, but investigation in this direction has not been considered so far. Another aspect we need to consider is the cross-layer design approach for the key management scheme. Previous studies assume that the communication in the lower layer is reliable, and the upper-layer protocols are designed so that there is no redundancy in the transmission of data packets. In the replacement of a group key, a node is not allowed to drop any of the data packets that are transmitted from the server to that node. This suggests that the retransmission of data packets will be requested everywhere across the network until all nodes receive all the required data packets through the unreliable wireless communication channel. If we add controlled redundancy to the upper-layer protocol, the issue of the retransmission of data can be completely mitigated.

The purpose of this paper is to refine and evaluate the group key management scheme previously proposed by Noda et al. [4] with a focus on the above observation. The proposed scheme consists of two components: an *attribute-based group structure* and a *group key revocation protocol*. The attribute-based group structure is a mathematical formalization of the family of groups in WSNs. The formalization is so flexible that it can model the wide variety of groups that are typically present in a WSN. The key revocation protocol is what controls the replacement of a group key. In basic terms, the server broadcasts the encryption of information that is needed to update the group key. The key point here is that the encrypted messages are composed in such a way that every legitimate node has a chance to receive multiple messages from which it can learn the required information, and the key is replaced successfully even if some of these messages get lost during the communication. We have already outlined the overall concept of our scheme in a preliminary study [4], but there are still many points that must be refined and substantiated. In this study, we describe the proposed scheme in detail and evaluate its efficiency under realistic conditions. The protocol has a controllable parameter that changes the redundancy of the transmitted data, and computer simulation shows that having some redundancy in the upper-layer protocol helps reduce the total amount of communication traffic.

II. Related Work

There have been quite a few studies that focus on the management of group keys, but not all of them can be used in WSNs. For example, there have been studies that exploit the flexible properties of public-key cryptography, but it is still arguable if public-key cryptography is acceptable for sensor nodes with limited resources. The self-healing mechanism has been considered for WSNs in [1], but it remains unclear if the computation over a finite field with a large order is feasible for sensor nodes.

In the following, we restrict ourselves to those schemes that are based on lightweight symmetric-key cryptography. A conventional work that conforms to this condition is widely known as the LKH scheme [7]. In this scheme, we consider a tree-like structure in which nodes are attached with key-managing keys (KEK) and in which leaves correspond to group members. The KEK at the root node plays the role of group key. The revocation of the group key is performed by constructing encrypted messages based on the tree structure. If there are n members in the group, the server broadcasts $O(\log n)$ different messages. There are many variations and extensions of the LKH scheme, but perhaps [3] is the most significant. In [3], we consider a scenario in which there are multiple groups in the network, and a node (user) belongs to one or more groups simultaneously. The mechanism in [3] mitigates the overhead for managing KEK, but the functionality of key revocation is degraded and we occasionally need to perform off-line reconstruction of key trees.

In the early days of WSN research, investigations were made to construct rudimentary but lightweight mechanisms for group key management. For example, in ZigBee [9], a *global-key* (network key) is embedded in all nodes in the network. This global-key can be regarded as the group key of a group that consists of all nodes, but we cannot use it as the group key of an "internal" group that contains only some of the nodes in the network. We should also point out that there is no explicit mechanism that helps revoke the global-key. We can use the node keys (master keys) of ZigBee and SPINS [5] to allow the server to send a group key to legitimate nodes, but such a protocol is essentially unicast-based and not efficient for large groups with many members. LEAP [8] is a powerful scheme that allows sensor nodes to form arbitrary groups. Its primary drawback is that all nodes in the network must have a precise timer and an apoptosis mechanism that diminishes critical information in the node, which seems to be an unrealistic expectation.

III. Attribute-Based Group

Generally speaking, a group is a set of nodes that have certain characteristics in common. Multiple groups can be defined in the network based on different characteristics, and a single node may belong to multiple groups in general. To establish a versatile model of such groups, we define a group structure in terms of *attributes* and *attribute values*.

An attribute is a characteristic that is associated with nodes. For example, "deploy location", "manufacturer", "type of equipped sensor", and "the most significant byte of the MAC address" are examples of attributes. For each attribute, a sensor node has a unique attribute value, where we assume that a special "undefined" attribute value is allowed if the attribute is not affiliated with a particular group of nodes. A group can be regarded as a set of nodes that have the same attribute values for certain attributes.

We can provide a mathematical formalization of the above intuitive definition. Let N be the set of all nodes in WSN.

Definition 3.1: An *attribute* is a set partition $A = \{G_1, \ldots, G_m\}$ of N, that is, m is a positive integer, $G_j \subset N$ for $1 \le j \le m$, $G_{j_1} \cap G_{j_2} = \emptyset$ for $j_1 \ne j_2$, and $G_1 \cup \cdots \cup G_m = N$.

We intend G_j to be the set of nodes that have the j-th attribute value for the considered attribute. Assume that there are d different attributes $A_1, \ldots, A_d$ in the network. We write $A_i = \{G_{i,1}, \ldots, G_{i,m_i}\}$ for $1 \le i \le d$, where m_i is the number of attribute values of the i-th attribute. The set of nodes $G_{i,j}$ with $1 \le i \le d$ and $1 \le j \le m_i$ is called a *base set*.

Definition 3.2: A set G of nodes is called an *attribute-based group* with *rank* r if G is defined as

$$G = G_{i_1,j_1} \cap \cdots \cap G_{i_r,j_r} \tag{1}$$

with $1 \le i_1 < \cdots i_r \le d$ and $1 \le j_c \le m_{i_c}$ for $1 \le c \le r$.

Note that base sets are attribute-based groups with rank 1.

In a practical WSN, a group is defined as a semantically meaningful set of nodes, while the attribute-based groups are sets of nodes that are defined mechanically from given attributes. This means that there can be many attribute-based groups that have little significance from a practical viewpoint. However, we strongly expect that semantically meaningful groups are also attribute-based groups if the attributes are chosen appropriately. For example, "the group of nodes deployed on the second floor" can be obtained by using "deploy location" as one of attributes and having "second floor" be one of the attribute values. It is possible to define four attributes corresponding to four bytes of IP(v4) addresses (with attribute values $\{0,\ldots,255\}$), and we can define "the group of nodes that belong to sub-net 192.1.2.*" as an attribute-based group with rank 3. We can use these attributes to consider an attribute-based group such as "the set of nodes that are deployed on the second floor and whose least significant byte of IP addresses is 123" while ignoring the meaningless attribute-based groups. In a sense, the attribute-based groups are a super-class of semantically meaningful groups. Therefore, we refer to attribute-based groups as simply *groups* in the following discussion.

For the practicality of discussion, we consider one additional condition for attributes, and assume that this condition is satisfied henceforth.

Definition 3.3: The set of attributes $A_1,\ldots,A_d$ is *complete* if no group with rank d contains two or more nodes.

This condition assumes that no two nodes have completely the same set of attribute values. This is quite a reasonable assumption because sensor nodes in the real world are all different in nature. Indeed, we can easily transform an incomplete set of attributes to a complete one by introducing an additional attribute that is based on device-unique identities such as a MAC address or that plays the role of a sequence number in a group. We should also point out that completeness implies that, for each node $n \in N$, there exists an attribute-based group G, such that $G = \{n\}$. The group key of G, which we define in the next section, can be used like the node key of ZigBee [9] owned by n and the server.

Example 3.1: Figure 1 shows an example of a complete set of attributes, where $N = \{n_1,\ldots,n_7\}$, and

$$G_{1,1} = \{n_1,n_2,n_3\}, G_{1,2} = \{n_4,n_5\}, G_{1,3} = \{n_6,n_7\},$$
$$G_{2,1} = \{n_1,n_4\}, G_{2,2} = \{n_2,n_5,n_6\}, G_{2,3} = \{n_3,n_7\},$$
$$G_{3,1} = \{n_1\}, G_{3,2} = \{n_2,n_4\}, G_{3,3} = \{n_3,n_5,n_6,n_7\}.$$

In the key revocation procedure, which is discussed in later sections, we need to consider a *set cover* of nodes. The set cover problem is a classical subject in computer science, but the set cover we need here is slightly different from the conventional one.

Definition 3.4: Let m be a positive integer and $n \in N$. A collection of base sets $\mathcal{G} = \{G_1,\ldots,G_l\}$ is an *(m,n)-cover* if

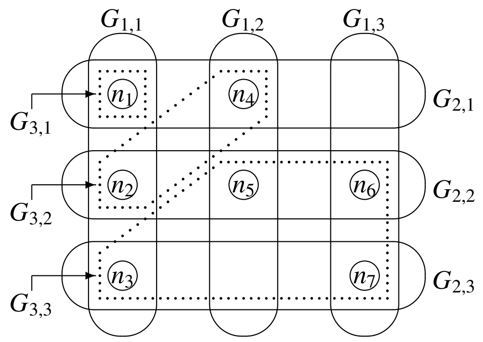

Figure 1. An example of a complete set of attributes.

the following conditions hold.

1) For each $n' \in N \setminus \{n\}$, $\mathcal{G}$ contains m or more base sets to which n' belongs.
2) For any i with $1 \le i \le l$, $n \notin G_i$.

Note that every node except n is covered at least m times by base sets in $\mathcal{G}$. The parameter m is called the *multiplicity* of the cover $\mathcal{G}$. Consider the attributes and groups that are given in Example 3.1. The collection of base sets $\mathcal{G} = \{G_{1,2}, G_{2,2}, G_{2,3}, G_{3,2}, G_{3,3}\}$ is a $(2,n_1)$-cover. For example, the node n_3 has two base sets $G_{2,3}$ and $G_{3,3}$ in $\mathcal{G}$ to which n_3 belongs. In the collection $\mathcal{G}$, there is no base set that contains n_1.

IV. Group Keys and Revocation Procedure

We have investigated secure and efficient group key management for attribute-based groups. In our framework, elementary information is distributed to sensor nodes, and group keys are then computed from this information.

A. Assumptions

First, we clarify the security assumption that is needed in the investigated framework.

The first assumption we need is that the compromise of a node is promptly detected by the server, which immediately reacts to revoke the compromised node. This condition is quite feasible in many WSN systems in which sensor nodes are watched by somebody, or equipped with a small physical security mechanism, and hence the condition does not interfere with the practicality of the procedure. We remark that this condition implicitly assumes that a malicious attacker does not compromise multiple nodes at one time, which is essential in the following protocol. We also assume that all nodes agree with a cryptographic hash function h and a symmetric-key cryptography. We write $E(k,x)$ to denote the result of the encryption of x using k as a key.

Each node has two tables of keys, called an *active-key table* and a *next-key table*. The former is used to store cryptographic keys that are currently active and used in the network, and the latter is used to store keys that will be used when the currently used keys expire. The active-key table is associated with the *version number* and the *expiration time*, and the next-key table is associated with the *version number*, the *activation time*, and the *expiration time*. The usage of these information will be explained later.

B. Group Keys

Assume that each base set is associated with secret information, which is called *base key*. The base key of $G_{i,j} \in A_i$ with $1 \le i \le d$ and $1 \le j \le m_i$ is denoted by $k_{i,j}$. The base keys are managed in such a way that a node knows $k_{i,j}$ if and only if that node belongs to the base set $G_{i,j}$. Group keys are defined by using base keys as follows.

Definition 4.1: Let G be an attribute-based group defined as (1). The *group key* of the group G is $k(G) = h(k_{i_1,j_1} \| \cdots \| k_{i_r,j_r})$, where "$\|$" is the concatenation of keys.

It is easily understood that a node is able to compute $k(G)$ if and only if that node belongs to the group G.

At the system initialization, the server determines a base key $k_{i,j}$ for each $1 \le i \le d$ and $1 \le j \le m_i$. The server also determines the *version number* and the *expiration time* of this set of base keys. The server delivers $k_{i,j}$ to all nodes in $G_{i,j}$ through a secure communication channel, together with the version number and the expiration time. The delivery of the base key and related information is performed in a safe place, possibly before nodes are deployed.

A node receives base keys and related information from the server and records them to the active-key table. At this time, the next-key table is empty. The key tables are managed by each node according to two principles:

- The content of the next-key table overwrites that of the active-key table when it gets to the activation time of the next-key table.
- If the active-key table is going to expire but the content of the next-key table has not been received yet, the node sends a NACK message to the server to request (re)transmission of the key information.

The server activates and expires base keys so that the keys are synchronized between the server and nodes. The server also receives NACK messages from sensor nodes and responds to nodes by sending requested information in a unicast manner, where the information is encrypted by using a group key that is available to the destination node.

C. Key Revocation

Group keys are replaced for two primary reasons. The first is that the keys tend to be used for too long a time. In general, it is not recommended to use a single key for very long because this gives attackers an opportunity to make a cryptanalysis and increases the risk of possible key leakage. From the security viewpoint, the periodical replacement of group keys is always recommended, even if there is no apparent security issue. The other reason for replacing a group key is the revocation of nodes. If a node in the network is compromised by somebody, we must consider a worst-case scenario, i.e., that an attacker has accessed all the information stored in the node. Cryptographic keys stored in the node are no longer secure, and we must replace them immediately. In such a case we need to deliver updated group keys to all nodes in the group except the compromised node. This procedure is called group key revocation. Generally speaking, the periodical replacement of group keys can be regarded as a special case of a group key revocation with no node revoked. In the rest of this section, we describe the group key revocation in detail.

Consider a scenario in which the server detects that a node $n \in N$ has been compromised by a malicious attacker. Without loss of generality, we assume that the node n belongs to d base sets $G_{1,1}, \ldots, G_{d,1}$ and has d base keys $k_{1,1}, \ldots, k_{d,1}$. We need to replace these d base keys because the attacker may discover them by disassembling the node n. The straightforward approach to solving this issue is to deliver a new base key (a replacement for $k_{i,1}$) to legitimate nodes in $G_{i,1} \setminus \{n\}$ for each of $1 \le i \le d$. However, to simplify the communication and to devise the cross-layer mechanism of the protocol, we consider a protocol in which the server sends a single "modifier" to all nodes other than n and enables nodes to replace their own base keys by using the modifier information. The modifier must be protected by encryption in such a way that it is accessible from all nodes other than n and that node n cannot learn what the modifier is. To achieve this requirement, the server performs the following procedure.

1) Determine the multiplicity parameter m and modifier string s. Also determine the version number v', the activation time a', and the expiration time e' of the updated set of base keys. The version number v' must be bigger than the version number of the currently used base keys.
2) Compute an (m,n)-cover $\mathcal{G} = \{G_{i_1,j_1}, \ldots, G_{i_l,j_l}\}$, where $1 \le i_c \le d$ and $1 \le j_c \le m_{i_c}$ for $1 \le c \le l$.
3) Broadcast l messages

$$M_c = (i_c, j_c, E(k_{i_c,j_c}, s \| v' \| a' \| e')) \qquad (1 \le c \le l). \quad (2)$$

Upon receiving the message M_c, a node n' performs the following procedure.

1) Discard the message if $n' \notin G_{i_c,j_c}$. Otherwise (i.e., $n' \in G_{i_c,j_c}$), proceed to the next step.
2) Decrypt the third component of M_c and retrieve s, v', a', and e'.
3) Do either one of the following operations.
   - If the next-key table is not defined, then record $h(k_{i,j} \oplus s)$ as the next base key of the base set $G_{i,j}$ with $n' \in G_{i,j}$ (and hence n' knows $k_{i,j}$),
   - If the next-key table is defined but its version number is smaller than v', discard the content of the table and perform the above operation.
   - In other cases, the next-key table is not modified.

In updating key information in the step 3, the hash function h is used to mitigate the risk of possible leakage of old keys. Without this hash function, an adversary who happens to know old keys may find the latest keys by investigating the XOR relation between new and old keys. Remark also that the update of the next-key table is controlled by the version number. This is to avoid possible confusion caused by the delay of message delivery, duplicated delivery of the same message, replay attacks by adversaries, and so on. As we explained previously, the next-key table replaces the active-key table when the appropriate time comes.

The focal point of the above protocol is that the messages M_c in (2) are determined from an (m, n)-cover. The server prepares messages in such a way that every legitimate node is given m or more messages that make sense to that particular node. In wireless communication, we cannot avoid the fact that some of these messages will be lost during the communication; however, we do not have to be too nervous about this because if just one of the m messages is delivered to a node, that node can successfully update its base keys. The multiplicity m controls the redundancy of transmitted messages, and the redundancy mitigates the effect of communication failure in the lower layer of communication.

V. Experiment

We performed preliminary experiments to determine how the multiplicity of a cover affects the total amount of traffic in a realistic setting. These experiments were preliminary in two senses. First, we consider the periodical replacement of group keys and ignore the node revocation scenario. This is because we need to introduce many additional parameters and assumptions if we would like to consider compromised nodes. The second reason we consider these experiments preliminary is that we do not discuss the network-wide burden of the protocol. We will evaluate two significant quantities related to the protocol, but will not consider other aspects of the protocol or the network. In this sense the experiment is limited, but this simple setting is effective in terms of concentrating on the cross-layer effect of multiplicity on the actual traffic.

We consider a WSN that contains 1,024 nodes with one of the nodes playing the role of the server. The nodes are deployed so that they form a square grid matrix of 32 $\times$ 32 nodes. The dimension of one unit grid is 8 m per side, meaning that the 1,024 nodes are deployed in a 248 $\times$ 248 m field. We assume that the node at the (x, y) position $(0 \leq x, y \leq 31)$ is given a *sequence number* $x + 32y$. The node with the sequence number 512 is deployed at almost the exact center of the field, and it plays the role of the designated server. We define ten attributes $A_1, \ldots, A_{10}$ with $A_i = \{G_{i,0}, G_{i,1}\}$ for $1 \leq i \leq 10$ in such a way that a node with a sequence number n belongs to $G_{i,j}$ if and only if the i-th most significant bit of the binary representation of the number n equals j. For example, the node with the number $858 = (1101011010)_2$ belongs to

$$G_{1,1}, G_{2,1}, G_{3,0}, G_{4,1}, G_{5,0}, G_{6,1}, G_{7,1}, G_{8,0}, G_{9,1}, G_{10,0}.$$

The nodes are placed in a grid matrix manner, so it is natural to consider that nodes in a row, or in a column, consist of one group, and that the network contains 64 groups in total. Such a group can be represented as an attribute-based group. For example, the group of nodes in the first row (i.e., $y = 0$) is defined as an attribute-based group of rank five $G_{1,0} \cap \cdots \cap G_{5,0}$, which contains nodes with numbers from $0 = (0000000000)_2$ to $31 = (0000011111)_2$.

In the preliminary experiment, we investigated the traffic when replacing the group keys for these 64 groups. We used QualNet for the computer simulation. The assumed protocols were IPv4 in the network layer and IEEE 802.15.4 in the MAC and PHY layer. The physical layer payload was 127 bytes, which is sufficient to contain the messages M_c in (2) in one packet. The wireless communication used a 2.4-GHz band with O-QPSK modulation. The communication speed was 250 kbps. The transmission power, antenna gain, and related parameters were adjusted so that the wireless range was almost equal to 14 m. Specifically, the transmission power was -17 dbm and the antenna gain was set to -3.0 dB. We employed the free space propagation model. The 14-m range allowed a node to communicate with eight neighbor nodes. A routing tree was manually provided, and the server and each node communicated with each other in a multi-hop manner. The server took 200 ms for the transmission time interval of packets, and nodes tried to avoid possible collision by using random jitter, where the jitter time was upper-bounded by various constants. For the control of broadcast messages in the network layer, nodes inspected the communication of their children with the passive ACK principle in which the waiting time was 150 ms. Retransmission of packets in the network layer was permitted up to three times.

At the beginning of the protocol, we set the key tables of nodes so that the active base keys expire in 60 seconds while the next-key table is empty. The server transmits messages for the periodical replacement of base keys and the nodes process the received information. After 60 seconds, nodes that could not obtain new base keys start sending NACK messages. A node sends one NACK message every second until it succeeds in updating its node keys. Upon receiving NACK messages, the server retransmits the modifier information to individual nodes that have sent NACK messages. The duration of the simulation was 120 seconds, which is sufficient for all nodes to finish replacing their base keys.

Figure 2 shows the number of packets that are transmitted by the server for the sake of key replacement, including packets transmitted as responses to NACK messages. The x-axis of the graph is the multiplicity m of the cover, and the y-axis is the number of packets. In the periodical replacement of group keys, the server initially transmits $2m$ packets,

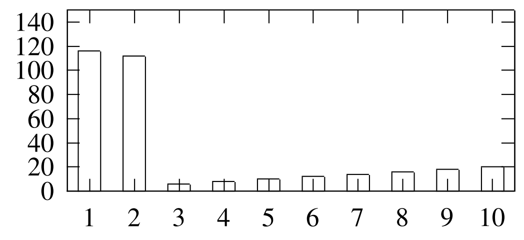

Figure 2. The number of packets transmitted by the server.

Table I
THE NUMBER OF NODES THAT ISSUE NACK MESSAGES.

Multiplicity m	NACK nodes	NACK messages
1	13	396
2	6	289
3–10	0	0

one for each $G_{i,j}$ with $1 \le i \le m$ and $j = 0, 1$. If every node receives one or more packets from which it can obtain modifier information, the retransmission of packets is not necessary. Such a favorable scenario is more likely if the multiplicity m is large, because large multiplicity means that nodes are given a greater chance of finding the modifier information. Indeed, the graph shows that the retransmission of packets is not needed for $m \ge 3$. On the other hand, if m is a small value, then some nodes fail to receive the required information. Such nodes send NACK messages to the server, and the server needs to send additional packets as responses to NACK messages. The graph clearly shows that having redundancy with controlled multiplicity helps decrease the total number of packets transmitted by the server.

Table I shows the other aspect of the protocol by showing the number of nodes that needed to issue NACK messages and the total number of NACK messages issued by nodes. If m was small, some nodes failed to receive the initial packets and had to send NACK messages to the server. We can see that one node issued several NACK messages, which implies that the NACK processing takes a rather long time. Detailed observation suggests that if a packet is lost near the server, many subsidiary nodes send NACK messages, which results in an unfavorable increase of communication traffic.

As above, adding some redundancy is good for reducing the total communication traffic in a realistic setting. We remark however that adding redundancy is not always as easy and efficient as in the attribute-based key management case. For comparison, consider that the LKH scheme is used to manage 64 group keys in the above experiment. In an idealized environment, the server sends 64 packets for periodical key replacement because only one packet is needed for each key tree. The observation in the above experiment suggests, however, that there will be 13 events in which a node cannot receive a packet. With 64 transmitted packets, the total number of such events is estimated to 832, and a large number of retransmission will be requested. The number of such events may be reduced by sending one packet m times, though, it increases the number of initially transmitted packets to $64m$. Remind that the number of initially transmitted packets is $2m$ in the attribute-based key management approach, and therefore, the overhead for increasing the multiplicity is more in LKH than the attribute-based key management.

VI. CONCLUSION

The cross-layer design effect in a group key management is discussed. It is shown that having controlled redundancy contributes to reduce the total amount of communication traffic in a realistic setting. This effect may exist in any key management protocols, but we saw that the overhead for having redundancy in the attribute-based key management scheme is smaller than that for widely known LKH scheme. From these results and observations, the attribute-based key management is concluded to be suitable for managing multiple group keys in large and practical WSNs.

ACKNOWLEDGMENTS

This work was partly supported by the Ministry of Internal Affairs and Communications (MIC).

REFERENCES

[1] R. Dutta, E. Chang, and S. Mukhopadhyay, Efficient Self-Healing Key Distribution with Revocation for Wireless Sensor Networks Using One Way Key Chains, 2007 Applied Cryptography and Network Security, pp. 385–400, 2007.

[2] B. Jiang and X. Hu, A Survey of Group Key Management, 2008 Intl. Conf. on Computer Science and Software Eng., Wuhan, China, pp. 994–1002, 2008.

[3] E. Jung, A. Liu, and M. Gouda, Key Bundle and Parcels: Secure Communication in Many Groups, Computer Networks, 50, pp. 1782–1798, 2006.

[4] J. Noda, Y. Kaji, and T. Nakao, A Group Key Management Scheme for Sensor Nodes Belonging to Multiple Large-Scale Groups, Journal of Information Processing, *52*, 3, pp. 1160-1172, 2011 (in Japanese).

[5] A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, and D.E. Culler, SPINS: Security Protocols for Sensor Networks, Wireless Networks, *8*, 5, pp. 521–534, 2002.

[6] S. Rafaeli and D. Hutchison, A Survey of Key Management for Secure Group Communication, ACM Computing Surveys, *35*, 3, pp. 309–329, 2003.

[7] C.K. Wong, M. Gouda, and S.S. Lam, Secure Group Communications Using Key Graphs, 1998 ACM SIGCOMM Conf. on Applications, Technologies, Architectures, and Protocols for Comput. Comm., pp. 68–79, 1998.

[8] S. Zhu, S. Setia, and S. Jajodia, LEAP: Efficient Security Mechanism for Large-Scale Distributed Sensor Networks, 10th ACM Conf. on Comput. and Comm. Security, pp. 62–72, 2003.

[9] IEEE 802.15 WPAN Task Group 4 (TG4), http://www.ieee802.org/15/pub/TG4.html ⟨10.04.2012⟩