Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2012, Article ID 301750, 14 pages
doi:10.1155/2012/301750

Research Article

A Secure Cluster Formation Scheme in Wireless Sensor Networks

Gicheol Wang,[1] Dongkyun Kim,[1] and Gihwan Cho[2]

[1] *Department of Advanced KREONET Service, Korea Institute of Science and Technology Information, Daejeon 305-806, Republic of Korea*
[2] *Division of Computer Engineering, Jeonbuk National University, Jeonju 561-756, Republic of Korea*

Correspondence should be addressed to Gihwan Cho, ghcho@chonbuk.ac.kr

Received 15 June 2012; Accepted 11 September 2012

Academic Editor: Mihui Kim

In wireless sensor networks, clustering expedites many desirable functions such as load balancing, energy savings, and distributed key management. For secure clustering, it is very important to find compromised nodes and remove them during the initial cluster formation process. If some nodes are compromised and survive the censorship process, they can make some nodes have a different cluster view and can split a cluster into multiple clusters to deteriorate cluster quality as a whole. To resolve these problems, we propose a robust scheme against such attacks in this paper. First, our scheme generates large-sized clusters where any two nodes are at most two hops away from each other to raise the quality of clusters. Second, our scheme employs the verification of two-hop distant nodes to preserve the quality of the large-sized clusters and refrains from splitting the clusters. Last, our scheme prefers broadcast transmissions to save the energy of nodes. Security analysis proves that our scheme can identify compromised nodes and preserves the cluster membership agreement against the compromised nodes. In addition, simulation results prove that our scheme generates fewer clusters and is more secure and energy efficient than the scheme producing only small-sized clusters.

1. Introduction

The recent advancement of wireless technology, sensor technology, and low-power embedded systems have enabled the development of low-power wireless sensing devices and they can collectively constitute a wireless sensor network for various applications. Wireless sensor networks have been widely deployed for military surveillance, pollution and structure monitoring, industrial equipment monitoring, mountain fire monitoring, and so on. Clustering in such networks provides many advantages such as the reduction of energy consumption [1, 2], load balancing [3], and distributed key management [4, 5].

The most prominent benefit of clustering is that it can greatly reduce the energy consumption of nodes and lengthen the network lifetime. Clustering is grouping physical network nodes into a small number of logical assemblies and maintaining them during the network operation. The logical assemblies are called clusters. For the initial formation of clusters, each node performs a cluster formation protocol. If each cluster requires a leader, nodes in each cluster should perform a leader election protocol. Hereafter, we call the leader as cluster head. Since a cluster head plays a crucial role such as collecting sensed data from other nodes and transferring the collected data to the sink, compromised nodes try to become cluster heads. In order to keep compromised nodes from being a cluster head, we can use two main strategies. First, we can identify compromised nodes and remove them during the initial cluster formation. If a compromised node survives the censorship process, it can easily obtain candidacy for being a cluster head. Therefore, removing the compromised nodes during the cluster formation is the first defense line for secure clustering. Second, we can keep the compromised nodes from predicting and manipulating results in cluster head elections and expediting their wins in the elections. This strategy is the second defense line for secure clustering. In this paper, we only focus on the first defense line as the second defense line was covered in [6–8].

Essentially, protecting the first defense line is very important for secure clustering. If the first defense breaks down, the second defense line is also put in danger and suffers from compromised nodes. To keep the first defense line from external attackers, thus far, several schemes using symmetric cryptography [9–11] were proposed. However, they cannot

keep compromised nodes from obstructing the operation of the protocols while they also prevent the participation of external attackers. Sun et al. proposed a scheme using the protocol conformity check and asymmetric cryptography [12]. It is effective at keeping the two types of compromised nodes from obstructing the operation of the protocol. However, this scheme operates with only small-sized clusters (i.e., cliques) and splits the cliques whenever a suspicious node is found in the cliques. This generates many clusters and the average size of clusters also decreases. Moreover, this scheme causes a lot of communication overhead to verify the protocol conformity of nodes.

To resolve the above problems, we propose a novel cluster formation scheme in this paper. First, our scheme generates large-sized clusters where the hop distance between any two members is at most two. Second, our scheme minimizes the separation of clusters by exchanging information between two-hop distant nodes. Last, instead of unicast transmissions, our scheme mainly employs broadcast transmissions to reduce energy consumption. Generating and maintaining large-sized clusters can provide many useful applications. First, a large-sized cluster has a long TDMA transmission schedule in which each node transmits its reading less frequently than the case of a short TDMA schedule. Therefore, a long TDMA schedule saves the amount of energy consumption at nodes, and consequently lengthens the network longevity. Second, generating large-sized clusters for a network shortens the hop distance between any two clusters and provides routing efficiency when nodes deliver their reading to the sink through multiple clusters. Last, a large-sized cluster enhances security if its CH-role node changes time after time. That is, if a CH-role node changes, the keys used for communication of the CH and its members also change accordingly. Since a large-sized cluster has more keys used for communication of the CH and its members than a small sized cluster, it guarantees higher confidentiality and integrity for their readings against malicious nodes.

The rest of this paper is organized as follows. Related work concerning secure cluster formation is briefly described in Section 2. Section 3 provides the network and threat model. The details of the proposed scheme are described in Section 4, and Section 5 provides the security analysis and simulation results. Finally, Section 6 concludes this paper.

2. Related Work

Heinzelman et al. proposed LEACH (Low-Energy Adaptive Clustering Hierarchy) in which sensors declare themselves as a cluster head according to a probability of being a cluster head [1]. Because cluster heads consume much more energy than normal nodes, this scheme attempts to extend the network lifetime by assigning cluster head roles to all nodes alternately. In this scheme, some nodes with a higher probability than threshold declare themselves as cluster heads and other nodes join in one of them. However, this scheme has no measure to protect the cluster formation.

F-LEACH protects the cluster head election in LEACH [9]. In this scheme, a node declares itself as a cluster head using common keys shared with the sink, and the sink authenticates the declaration message using the same keys. Then, the sink securely broadcasts the authenticated cluster heads using μTESLA [13]. Normal nodes only join one of the authenticated cluster heads. However, this scheme has no mechanism that authenticates the joining of normal nodes. Oliveira et al. proposed SecLEACH [10] in which the sink authenticates the cluster head nodes and the cluster heads authenticate the joining nodes. In F-LEACH and SecLEACH, some keys for the authentication are distributed to sensors prior to deployment. However, both F-LEACH and SecLEACH can only protect the cluster formation from external attackers. In other words, they cannot prevent compromised nodes from declaring themselves as cluster heads and from joining in any cluster head.

Liu proposed a cluster formation scheme where only predetermined nodes declare themselves as cluster heads and other nodes join in any cluster directly or via a relay node [11]. Since the declaration of any cluster head or the join of any normal node is authenticated by preassigned polynomial shares, an external attacker cannot participate in the cluster formation. This scheme also has a wormhole prevention mechanism in which a node with many neighbors shuts itself down or the sink enforcedly shuts down a node with many neighbors by reporting the node as a wormhole attacker. However, if any relay node is compromised by attackers, the relay node can invoke a DoS (Denial of Service) attack by cutting down the connection between the cluster head and its serving nodes. In addition, a compromised node can disturb a relay node determination and break all connections using the relay node. Moreover, attackers can target the predetermined cluster heads for compromise because their roles are fixed.

Sun et al. proposed a secure cluster formation scheme which verifies the protocol conformity of nodes to identify malicious nodes [12]. In this scheme, all nodes are grouped into cliques, in which all nodes are directly connected with each other. After the clique formation, each node checks whether all members in the clique agree on the clique membership or not. If a normal node finds a disagreement, it performs the protocol conformity verification for other nodes in the clique in order to recognize and remove compromised nodes. This scheme well identifies and removes compromised nodes through the protocol conformity check. However, the scheme increases the number of clusters in the network because it only produces small-sized clusters (i.e., cliques) and splits a cluster whenever a suspicious node is identified in the cluster. Moreover, it burdens nodes with a lot of communication overhead because it requires an abundance of unicast communication during the protocol conformity check. To elaborate on why the unicast transmission causes more overhead than broadcast communication, we assume that a node transmits a message to its all neighbors serially. If we use unicast communication, the node should transmit the message as frequently as the number of the neighbors. Contrarily, the broadcast communication brings the same effect through only one-time transmission.

Rifà-Pous and Herrera-Joancomartí divided the cluster formation process into three phases; cluster discovery phase,

cluster head designation phase, and cluster maintenance phase [14]. In the cluster discovery phase, members in a cluster build a consensus in the cluster membership. In the cluster head designation phase, members in a cluster elect a cluster head based on the number of neighbors and the frequency of cluster-head-role performance. In the cluster maintenance phase, the elected cluster heads play the role of a local CA (Certification Authority), and issue an authorization certificate to each member. However, this scheme assumes that all nodes conform to the cluster discovery protocol. For instance, if a compromised node transmits a message to some nodes and avoids the transmission to other nodes in the cluster discovery phase, the nodes in the same cluster have a different membership. This separates a cluster into multiple ones, and the separated clusters elect their cluster head, respectively, in the cluster head designation phase. So, this scheme is vulnerable to such an attack and can generate many clusters in the network.

3. Network and Threat Model

3.1. Network Model. We assume that sensor nodes are randomly scattered in the work field by an aircraft. Following the deployment, they never change their locations and are grouped into clusters to perform an energy-efficient operation like TDMA communication in clusters. The sink acts like a data collection center and a gateway to the wired networks. We assume the following.

First, any wormhole attack is invalidated by a wormhole prevention scheme such as the scheme in [15] so that each node can identify its neighbors correctly. Second, there is no message loss during the cluster formation process except for the intentional transmission avoidance of compromised nodes. Even though it seems to be quite an immoderate assumption, we need this assumption to discriminate a compromised node from normal nodes. Without this assumption, normal nodes cannot discriminate the misbehavior of compromised nodes from message loss. Removal of this assumption is an interesting future research item. Third, we assume that each node can adjust its transmission power when they send a message so that at most two-hop distant nodes can receive the message. Last, each node can support lightweight public key operations such as ECC (Elliptic Curve Cryptography) operations. It has been proven in [16] that energy-constrained sensors can well support lightweight public key operations such as ECDSA (Elliptic Curve Digital Signature Algorithm) signature generation and ECDSA verification.

3.2. Threat Model. Numerous attacks are available in wireless sensor networks. One of the most serious attacks is the DoS attack. Even though this sort of attacks is very stealthy and difficult to defeat, some promising countermeasures were introduced in [17]. Especially in a cluster structure, a greedy attacker might affiliate many normal nodes into its cluster by transmitting a bogus and long-distant message. Thanks to a wormhole prevention scheme, our scheme can easily defeat this type of attack. Sybil attack [18] also has a bad impact on the network because nodes cannot identify their legitimate neighbors. We assume that such an attack can be paralyzed by the schemes in [19].

To focus on the secure cluster formation problem, we assume that compromised nodes launch an attack by not conforming to a cluster formation protocol. Especially, we concentrate on two kinds of attacks on a cluster formation protocol. First, a compromised node can transmit a message to a section of nodes in a cluster while avoiding transmission to the rest of the nodes. Hereafter, this type of attack will be termed "selective transmission attack." In addition, a compromised node can utterly avoid the transmission of a message. We term this type of attack as "silence attack" hereafter. These attacks force nodes to have a different view on the cluster membership. This splits a cluster into multiple ones and consequently decreases the average size of clusters (i.e., the average number of members). In a cluster, the number of members affects the probability that a compromised node is elected as a CH on the basis of random selection. To elaborate on this problem, we assume that there is one cluster with a few members and another cluster with more members and they have one compromised node, respectively. We also assume that a cluster head is elected randomly like in [6] and compromised nodes cooperate with the election protocol. In this situation, the cluster with a few members is likely to elect the compromised node as a cluster head.

4. Secure Cluster Formation Using Two-Hop Conformity Verification

Before the detailed explanation of our scheme, we first establish some definitions, which are used throughout the paper.

(i) *Dominants*: At the beginning of the cluster formation, each node exchanges its certificate with neighbors. Due to the exchange, each node comes to know the neighbors whose IDs are lower than its own ID. Those nodes are referred as *dominants* hereinafter and their IDs are maintained in each node's storage.

(ii) *CS (Cluster Separator) node*: A node whose ID is the lowest among the neighbors becomes a *CS node*. In the same manner, a node expects that the lowest ID node among the dominants is going to declare itself as a *CS node*.

After the deployment of sensor nodes, each node exchanges its certificate with neighbors. This exchange enables each node to identify its neighbors and dominants. The lowest ID node in the neighborhood broadcasts a CS (Cluster Separator) message to determine a cluster border. Note that a CS node is not a cluster head but a protocol initiator. If the receivers of the CS message do not belong to any clusters, they join the cluster and respond with a broadcast message to inform the CS node of their joining. Hereafter, we call this message as a CR (Cluster Response) message. If a sender of a CR message is a dominant, the receivers remove the sender from their dominant list and check if there is no dominant. A node having no dominants becomes a CS node. If there

is no attack on the protocol, this process creates clusters in the network incrementally. After the border determination of clusters, CS nodes broadcast a FCS (Final Cluster Separator) message to request members to accept their joining the cluster. This message contains a list of CR messages, which were sent from its cluster and it enables the comparison of the same membership between nodes. A receiver accepts the request to join only if the received membership is exactly same with its own membership. If a CS node attempts to exclude a specific node, it might prevent the propagation of the victim's CR message and omit the victim's CR message in its FCS message. When the victim notifies this misbehavior, the victim searches a node which holds the victim's CR message to be a witness by sending a Solicitation message. Such a node signs the victim's CR message with its private key and sends it with a pre-received FCS message to the victim. This message is called the Solicitation Response message hereafter. Through this message reception, the victim acquires a witness to prove its legitimacy. Because the CS node's misbehavior has been confirmed, the victim now accuses the CS node with an Attacker Report message for other members to exclude it. Table 1 shows the time when the messages are sent and what functions they do.

The cluster formation is divided into two steps. Step 1 determines the border of clusters and verifies it. Step 2 merges a CS node into its cluster and verifies the merge.

4.1. Cluster Border Determination and Verification

4.1.1. Cluster Border Determination. A CS node broadcasts a CS (Cluster Separator) message to initiate the cluster formation. The message consists of the message type and the sender ID, which is signed by the sender's private key to prevent a spoofing attack. If a node receives the message from multiple CS nodes, the receiver joins the first comer and ignores other messages.

If a node receives a CS message, it verifies the received signature. If it is correctly verified, it joins the cluster and broadcasts the CR (Cluster Response) message. The CS message and the CR message determine the border of a cluster. The CR message consists of the message type, the CS node's ID, and the received signature from the CS node. The node also signs the message with its private key before broadcasting to protect the integrity and the authenticity of the CR message. Last, the sender's certificate is attached to the tail of the signed message because it might be propagated further than one hop. Note that each node holds only neighbors' certificates. A node in the same cluster verifies the CR message and stores the message if it is correctly verified. Note that a CS node in the same cluster uses the stored CR messages as a proof when it requires the members to accept its merge into the cluster in step 2. The receiver of a CR message first checks whether the message comes from a member of the same cluster or not. If the message originator belongs to the same cluster, and it is the first message from the originator, the receiver rebroadcasts the message to prevent silence attacks. However, if the CR message comes from another cluster, the receiver does nothing except check whether they are dominants or not. In case of dominants, the receiver removes the sender from the dominants and it becomes a CS node if the dominant list is empty. Figure 1 illustrates the flowchart of cluster border determination in step 1.

4.1.2. Verification of Cluster Border Determination. After the cluster border determination, each node checks if there are any deviations from the protocol. If some deviations are found, each node employs the following actions.

Attack Type 1. If a CS node avoids the broadcast of a CS message, members in the cluster cannot receive any message from its cluster. In this case, the members remove the CS node from dominants and the neighbor list and check if there is no dominant. If so, they become a CS node and broadcast a CS message. Figure 2 shows this countermeasure following the "yes" control flow from the comparative symbol "No message from same cluster."

Attack Type 2. when a CS node is the sole link point among nodes in a cluster, it may avoid rebroadcasting a CR message from a node. In that case, the nodes which are connected via only the CS node cannot receive the CR message and those nodes have a different view on cluster membership. To agree on the cluster membership between those nodes, we employ the following countermeasures. Figure 2 shows these countermeasures following the control flow from connector 3.

(i) If a node receives no CR messages from any nonneighbors, the node broadcasts its CR message with two-hop transmission power. Then, it waits for unknown nodes to respond with their CR message. When the node receives a CR message from an unknown node (i.e., a nonneighbor node) in the same cluster, it registers the sender into the member list.

(ii) If a node has any CR message from a nonneighbor node, it waits for other CR messages from other nonneighbor nodes. When it receives a CR message from another unknown node, it registers the sender into member list and broadcasts its CR message with two-hop transmission power to notify its existence.

Attack Type 3. If a CS node broadcasts its CS message to only a part of its neighbors, some neighbors cannot receive the CS message but can receive a CR message. In that case, the nonreceivers of the CS message can recognize that the CS node selectively transmits the CS message. If a node receives only CR messages not a CS message, the node has two choices such as the following.

(i) First, the node can request other nodes in the same cluster to accept its affiliation. Because the requested nodes cannot assure if the requestor is lying or the CS node is a malicious node, the situation is very ambiguous. So, they can remove the requestor and the CS node from the cluster to resolve the ambiguity.

Table 1: Function of messages and the time when they are sent.

Message type	Time when message is sent *Function*
CS (Cluster Separator) message	At the beginning of step 1 *Determines a cluster border*
CR (Cluster Response) message	Upon receiving a CS message *Request to join a cluster*
FCS (Final Cluster Separator) message	At the beginning of the step 2 *Merges a CS node into a cluster*
Solicitation message	When a victim does not receive a FCS message from its CS node *Acquire witnesses to prove its legitimacy*
Solicitation response message	When a node receives a Solicitation message and holds any evidence *Proves a victim's legitimacy*
Attacker report message	When a victim confirms misbehavior of a CS node *Excludes a CS node from members*

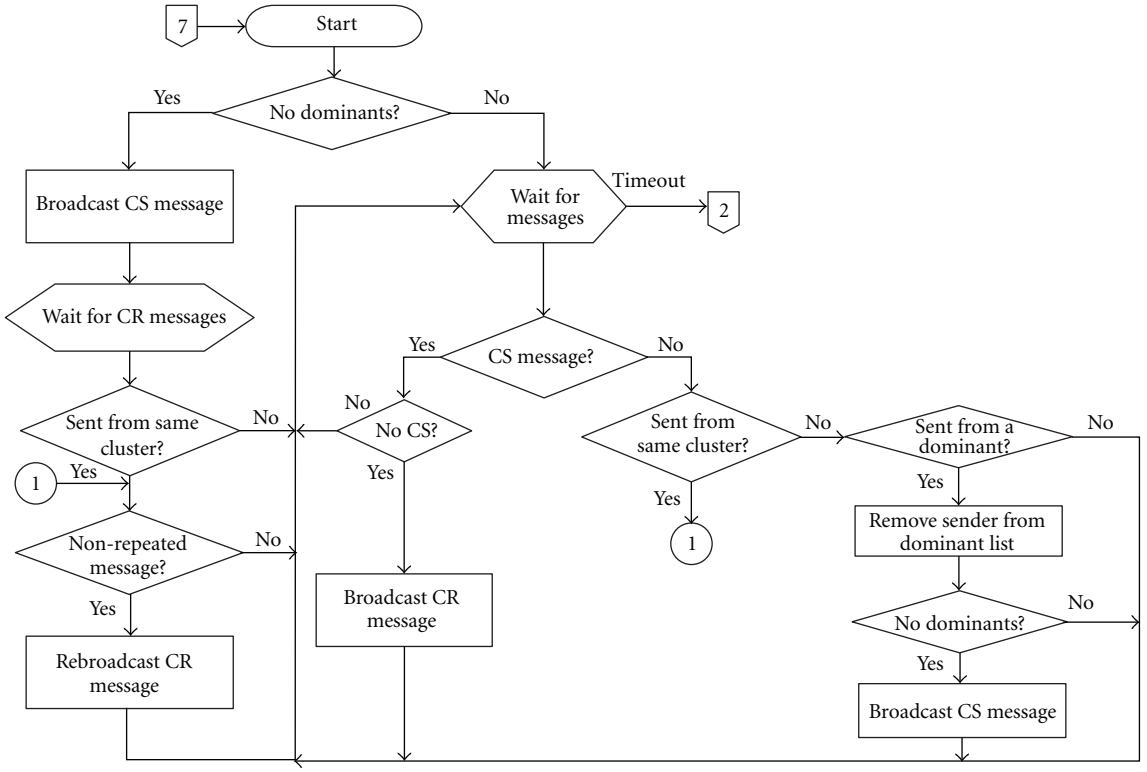

Figure 1: Flowchart of cluster border determination in step 1.

Since they are now separated from the original cluster, they need to restart the cluster formation. If the CS node is the real compromised node, it is likely to broadcast the CS message at this time. Otherwise, it is going to be separated alone.

(ii) Second, the node can remove the CS node from the member and neighbor list and restart the cluster formation to define a new cluster. In this case, because the non-receiver of the CS message does not need to send an affiliation request message, it can save its energy. If the CS node is the real compromised node, the receivers of the CS message have a malicious CS node in their cluster.

Regardless of taking any choice, the cluster is separated into two and one of them has a malicious CS node. Therefore, we select the second choice in order to save energy consumption. Following the second choice, some members might lose their CS node. If a node loses the connection with

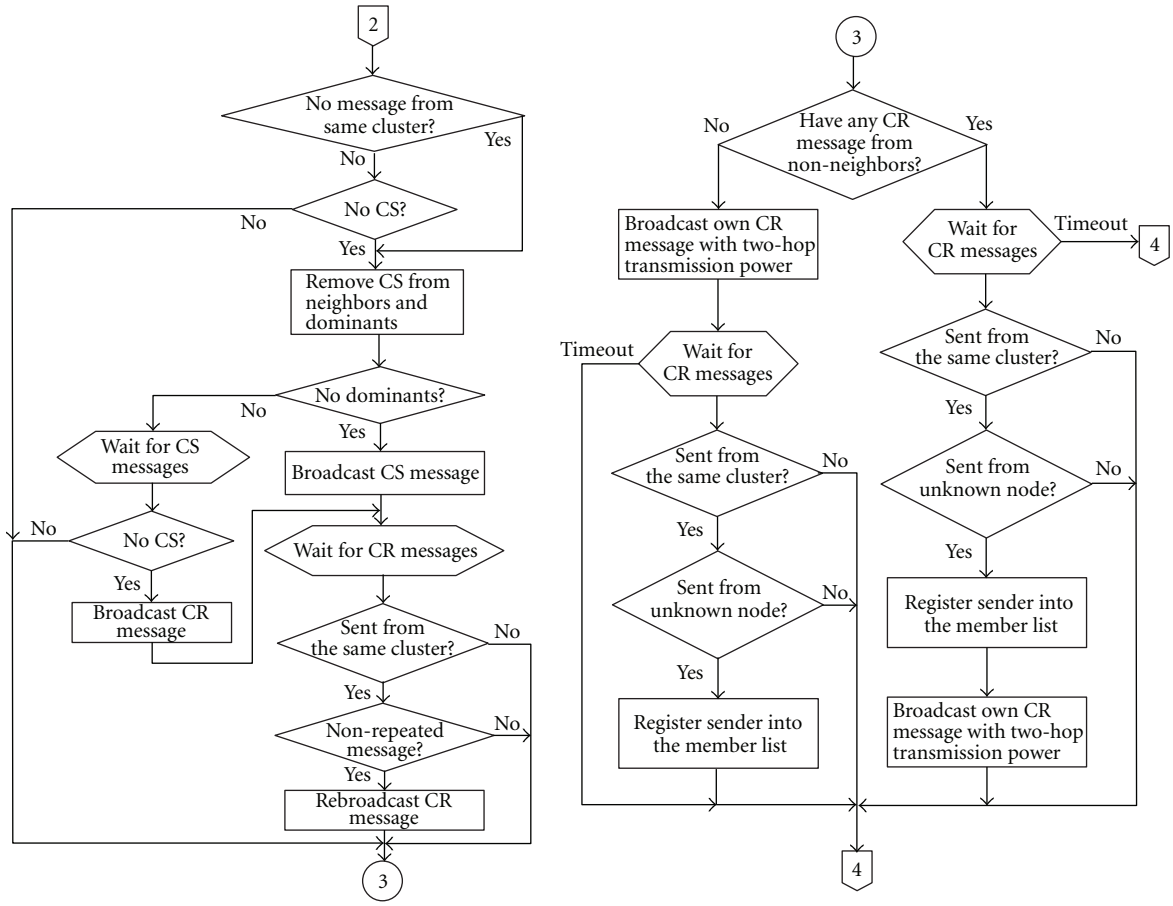

Figure 2: Flowchart of cluster border verification in step 1.

its CS node, it checks if there is no dominant in the neighborhood. If so, the node becomes a CS node and broadcasts a CS message. Figure 2 shows this countermeasure following the "no" control flow from the comparative symbol "No message from same cluster."

4.2. Final Cluster Formation

4.2.1. Merger of CS Nodes. A CS node broadcasts an FCS message using the received CR messages. The message consists of the message type and the list of the received CR messages. The CS node signs the message with its private key and transmits the signed message. Upon receiving an FCS message, the receiver verifies the signature and compares the list of CR messages with its own list. If they are exactly the same, the receiver registers the sender into the member list. Otherwise, the receiver discards the message.

4.2.2. Verification of Merged CS Nodes. After the merger of a CS node, each node checks if the CS node deviates from the protocol. If it identifies a deviation, it takes the following actions.

Attack Type 4. A CS node may remove some CR messages from its FCS (Final Cluster Separator) message and avoid the transmission to the removed nodes to exclude them from the cluster. In this case, the nonreceivers of the FCS message can employ the following countermeasures. Figure 3 shows the flowchart of the merger and verification of the CS node.

(i) If a node does not receive an FCS message from its CS node, it broadcasts a Solicitation message with two-hop transmission power.

(ii) A node which receives a Solicitation message checks if it has the sender's CR message. A compromised node may hide any misbehavior of the CS node by not responding to the Solicitation message even if it has the solicitor's CR message (Attack type 5). Otherwise, it makes a Solicitation Response message and responds to the solicitor with the message. First, it signs the solicitor's CR message with its private key and transmits the signed message in a unicast manner along with the FCS message and its certificate.

(iii) If a solicitor receives a Solicitation Response message, it first checks the FCS message, which is included in the message. The solicitor checks whether its CR message exists in the list of CR messages or not. Recall that a FCS message consists of CR messages received from members. So, if a FCS message misses the solicitor's CR message, this is unavoidable proof

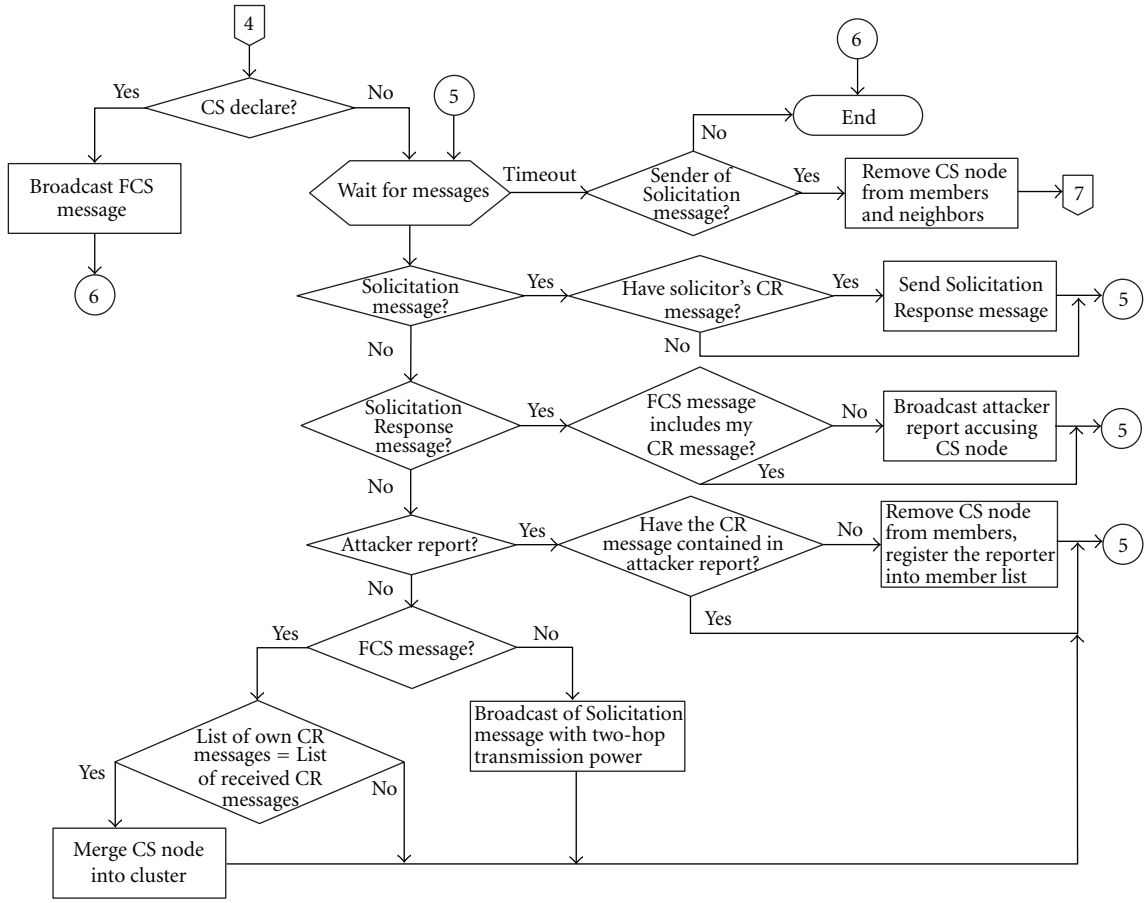

Figure 3: Flowchart of merger and verification of CS node.

that the CS node is a compromised node. In that case, the solicitor advertises the CS node as an attacker using a two-hop broadcast message, which is called an Attacker Report. The message consists of the signed CR message and the signer's certificate. If a solicitor is compromised, it may omit the transmission of the Attacker Report to avoid the exclusion of the CS node (Attack type 6).

(iv) If a node receives an Attacker Report, it verifies the message and checks if the witness really has received the CR message of which it proclaimed the reception. If so, it removes the CS node and registers the reporter into the member list. As a matter of fact, the node which receives the Attacker Report cannot assure that the accused CS node is really responsible for the nonreception of a CR message at any node. However, in any case, because the CS node is connected to all nodes in the cluster, it is most responsible for the nonreception.

(v) If a non-receiver of a FCS message receives no message even after it broadcasts a Solicitation message, it removes its CS node from the member and the neighbor list and restarts the protocol from the beginning.

4.3. Cluster Formation Example. We introduce an illustrative example to help the quick comprehension for our scheme. In step 1, nodes determine their cluster border. Figure 4(a) shows the initial process for the cluster border determination in step 1. Nodes 1, 4, and 5 become a CS node because they have no dominants in their neighborhood. So, they broadcast a CS message. However, a CS node (i.e., 5) deviates from the protocol by selectively transmitting its CS message. Node 5 does not send its CS message to nodes 9 and 27 to exclude them from the cluster. Nodes 9 and 27 cannot assure whether CS node 5 has deviated from the protocol or not until they receive a CR message heading for node 5 from another node in the cluster. Such a CR message can be a proof that CS node 5 broadcasted its CS message.

Figure 4(b) shows that each node receiving a CS message broadcasts a CR message in step 1. If a node receives a CR message and it is not a duplicate message, it rebroadcasts the message. However, a malicious node may avoid rebroadcasting the message. Node 4 carries out such an attack to hide nodes 30 and 40 from 29 and vice versa. To defeat this kind of attack, node 29 transmits its CR message with two-hop transmission power to find a hidden member because it does not receive a CR message from any two-hop member. Nodes 30 and 40 register the node 29 into their member list and broadcast their own CR message with two-hop transmission

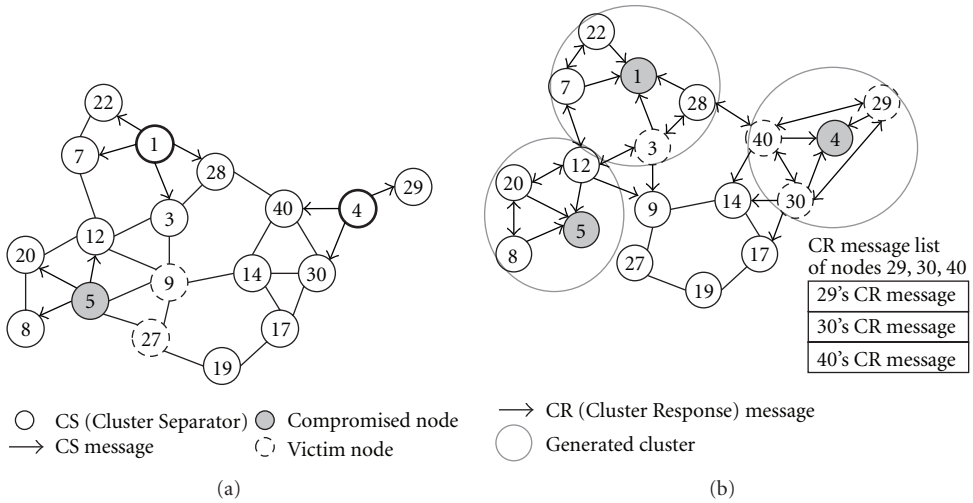

Figure 4: Cluster border determination in step 1 (a) Broadcast of CS message (b) Broadcast of CR message.

power. Node 29 also registers the nodes 30 and 40 into its member list. As a result, they share the same cluster mem-bership among them as shown in the right bottom of Figure 4(b).

Node 1 rejected the relay of the CR message of node 3 in order to exclude the node 3 from its cluster in Figure 4(b). However, the neighbor node 28 can get the CR message because it is directly connected with the node 3. As a result, the nodes 7 and 22 have a different view on cluster membership from nodes 3 and 28 as shown in Figure 5(a). Even though nodes 9 and 27 do not receive any CS message, they can receive a CR message from node 12 as shown in Figure 4(b). So, they can assure that CS node 5 avoided the transmission of its CS message to them. Therefore, they remove CS node 5 from their member list and neighbor list. Now, they restart the cluster formation process to define a new cluster border. Since node 9 no longer has any dominants, it declares itself as a CS node as shown in Figure 5(a). Upon receiving the CS message, nodes 14 and 27 broadcast their CR message to join the cluster as shown in Figure 5(b). Node 17 performs the same process as node 9 and the result is shown as Figure 6(a).

In step 2, nodes merge the CS node into their cluster if it conforms to the protocol. CS nodes trigger step 2 as in step 1. Nodes 1, 4, 5, 9, and 17 initiate step 2 by broadcasting an FCS message as shown in Figure 6(a). Since compromised nodes 4 and 5 behave as if they are normal nodes in Figure 6(a), we concentrate on the absorption of CS node 1. Recall that CS node 1 avoided rebroadcasting node 3's CR message in step 1 in order to separate it from the cluster. At the beginning of step 2, CS node 1 broadcasts its FCS message including the received CR messages. Of course, the CS node omits node 3's CR message to cheat other nodes and avoids the transmission toward node 3 as shown in Figure 6(a). Receivers 7 and 22 compare the list of CR messages with their own list. Because nodes 7 and 22 discover that they are exactly the same, they register node 1 into their member list. However, node 28 does nothing because it discovers that node 1's CR message list disagrees with its own list. Now, node 3 broadcasts a Solicitation message with two hop transmission power to obtain proof that it broadcasted its CR message as shown in Figure 6(b). Since receiver 28 has node 3's CR message, it first signs node 3's CR message by its private key and transmits the signed message along with 1's FCS message. The signed CR message of node 3 and the FCS message of node 1 constitute a Solicitation Response message as shown in Figure 6(b).

Upon receiving the Solicitation Response message, node 3 checks if there are unknown nodes in the FCS message. If so, it registers those nodes into its member list. Then, node 3 checks whether node 1's FCS message includes its CR message or not. Since node 1's FCS message does not include its CR message, it is definite proof of node 1's deviation from the protocol. So, node 3 reports node 1 as an attacker using a two-hop broadcast message as shown in Figure 7(a). The Attacker Report includes node 3's CR message which is signed by node 28's private key and node 28's certificate. Receivers 7, 22, and 28 verify the signature. If the verification succeeds, they remove node 1 from the member list and the neighbor list. This is because node 1 is a CS node, and it is connected to all nodes. That is, it is most responsible for the nonreception of 3's CR message at nodes 7 and 22. Nodes 7 and 22 register node 3 into their member list because they find a new normal node whose normalcy is guaranteed by node 28.

Now, all processes for cluster formation are completed. We have a clustered network such as Figure 7(b) after the completion of the protocol.

5. Evaluation

5.1. Security Analysis. Our scheme prevents external attackers from joining the cluster formation process by using message authentication. For such a reason, we focus on the insider attacks, which are launched by compromised nodes. If compromised nodes transmit the same false message or

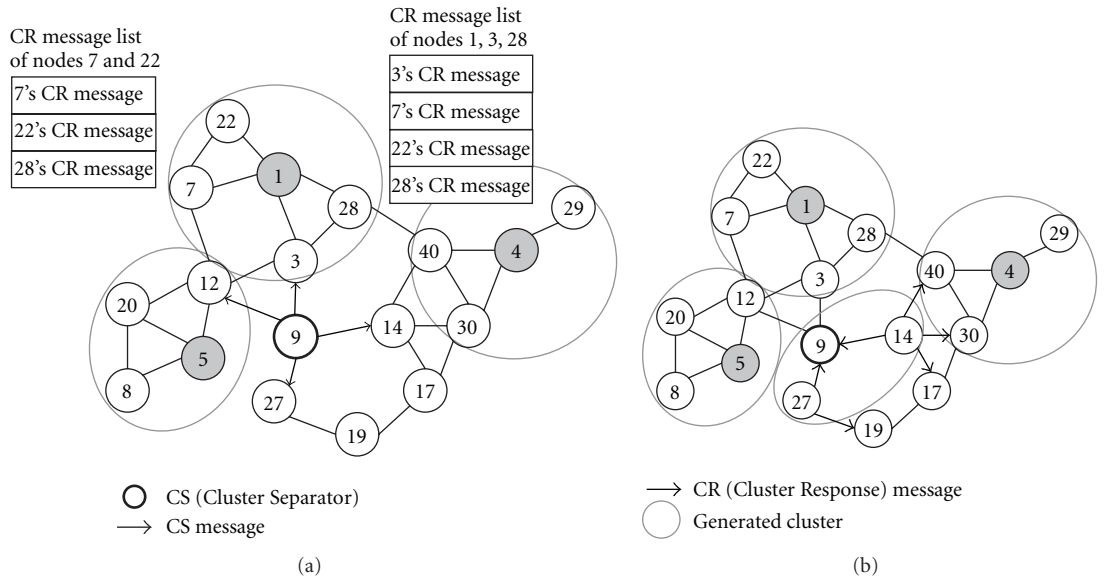

Figure 5: New cluster border determination in step 1 (a) Disconnection from existing CS node and new cluster border determination (b) Broadcast of CR message for new CS node.

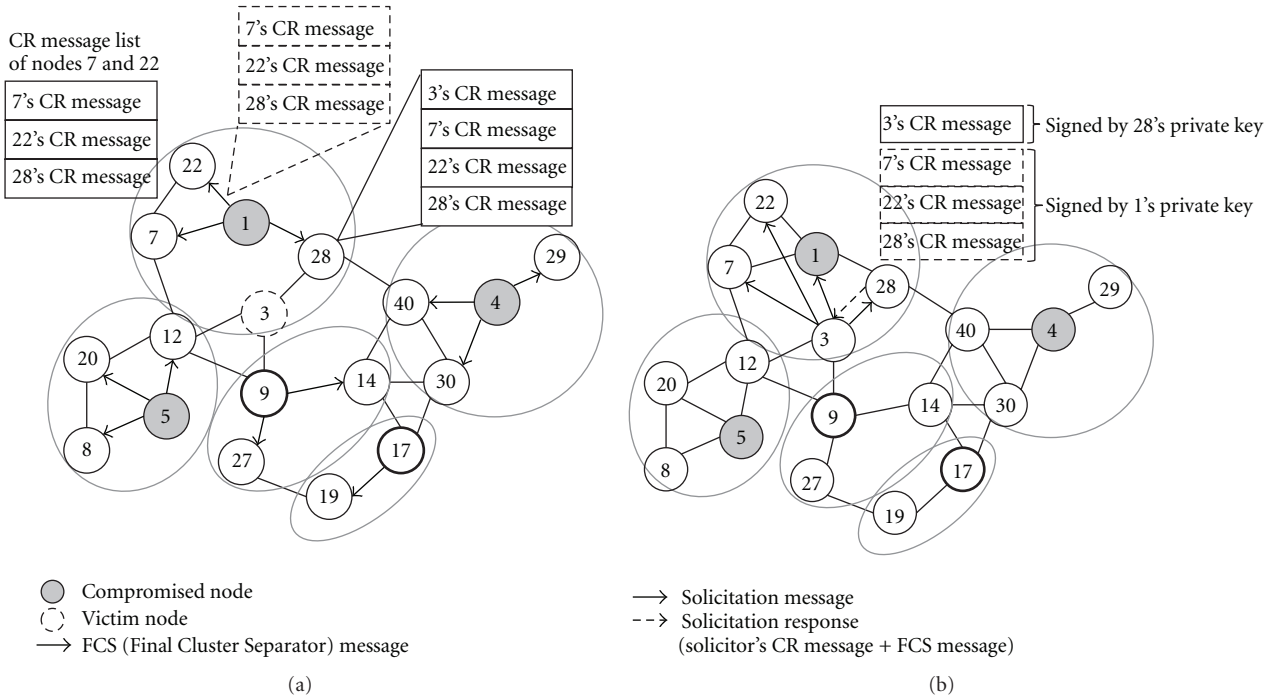

Figure 6: Merger of CS node and unknown normal nodes in step 2 (a) Broadcast of FCS message (b) Solicitation request and response.

avoid transmission to all members, they cannot create any inconsistency in the cluster membership. Therefore, any inconsistency in cluster membership can only result from transmitting different messages to different members or from avoiding the transmission by compromised nodes. In Section 5.1.1, we prove that compromised nodes are identified if they transmit inconsistent messages. In Section 5.1.2, we prove that our scheme preserves the agreement in cluster membership against the misbehavior of compromised nodes.

5.1.1. Identification of Compromised Nodes. First, we explain notations employed in the following. Suppose that nodes i, j, and k are normal members in a cluster and node l is the CS node in the cluster. We denote node i's view on cluster membership as Ci in the following. We introduce Lemma 1 and employ it to prove Theorem 2.

Lemma 1. *For two normal members i and j, if $k \in Ci$, then we should have $k \in Cj$.*

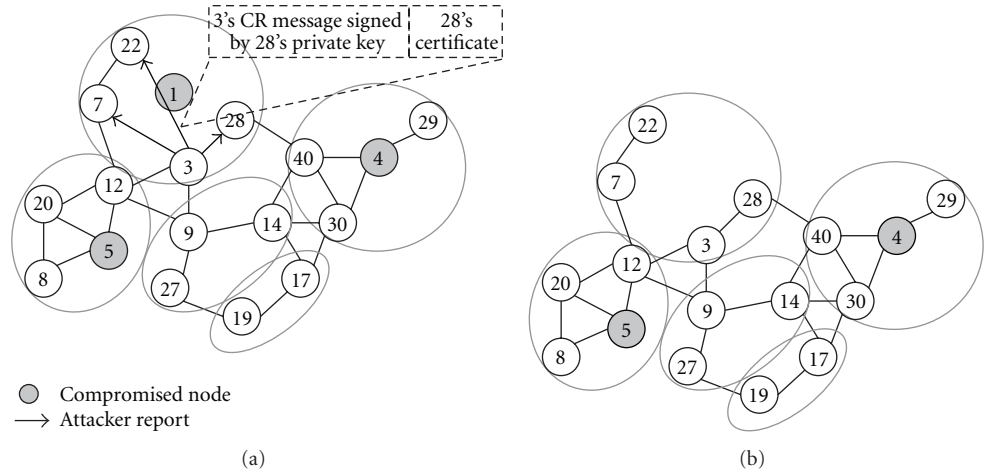

Figure 7: Attacker report and completion of cluster formation (a) Attacker report (b) Completion of cluster formation.

Proof. We prove it by contradiction. First, we suppose $k \notin Cj$. If the supposition is true, we should have $k \notin Cj$ in the first step of our scheme. If $k \notin Cj$ in the first step of our scheme, we also have $k \notin Cj$. Therefore, it contradicts the condition $k \in Ci$. □

Theorem 2. *If any inconsistency in cluster membership is caused by compromised nodes which transmit inconsistent messages to members, our scheme can identify the compromised nodes.*

Proof. As shown in Lemma 1, if $k \in Ci$, then we have $k \in Ci$. Therefore, if we identify any inconsistency in cluster membership, it has been definitely caused by the CS node l which is connected to all members i, j, and k. When the node k identifies inconsistency in cluster membership, the node k transmits a Solicitation message and node i responds to the Solicitation message. Using the Solicitation Response message, we come to know that the CS node l transmits inconsistent messages to members. □

5.1.2. Preservation of Cluster Membership Agreement. In our scheme, the silence attacks can only cause any inconsistency in membership in the second step. First, if a CS node causes any inconsistency in membership during the first step, the victims are separated from the original cluster. Since the victims form a cluster for themselves, the separation and the formation cannot affect the final cluster membership. Second, if some compromised nodes do not relay CR messages in the first step, members themselves agree on the cluster membership with each other through a two-hop broadcast of the CR message. In the following Theorem 3, we prove that our scheme preserves the agreement of cluster membership even if compromised nodes launch a silence attack or a selective transmission attack to certain members.

Theorem 3. *For a CS node l and two normal members i and j, if $l \in Cj$, then we must have $Ci = Cj$.*

Proof. We prove it by contradiction. First, we suppose that $Ci \neq Cj$. Without loss of generality, we assume that $k \in Ci$ but $k \notin Cj$. In this case, node j identifies the CS node l's misbehavior through the node k's Attacker Report. Then node j removes the CS node l from its member list and registers the node k into its member list. Therefore, we have $l \notin Cj$. It is contrary to the condition $l \in Cj$. □

5.2. Simulation Results. In this subsection, we evaluate the security and energy-efficiency of our scheme using the ns-2 network simulator. In the simulation environment, 100 nodes were randomly deployed in a 100 m × 100 m square field. Each node consumed their energy using the energy model of [1] during the cluster formation. Table 2 shows the simulation parameters and their values. We compare our scheme with Sun's scheme [12] because its goal and method are most similar with our scheme. Even though some schemes [9–11] present their secure cluster formation techniques, they are excluded from comparison because their methodology and attackers' aim are significantly different from our scheme. Rifà-Pous' scheme is similar to our scheme in terms of cluster formation methodology but it has no countermeasures against compromised nodes which deviate from the protocol. So, comparison with Rifà-Pous' scheme is unfair.

In Figure 8(a), we demonstrate how many clusters the two schemes generate when compromised nodes launch silence attacks and selective transmission attacks on the network. The result shows their resiliency when they are under attack. As shown in Figure 8(a), both schemes increase the number of clusters as the number of compromised nodes increases. Our scheme greatly reduces the generated clusters as compared with Sun's scheme, especially under a small number of compromised nodes. This result is caused by two facts. First, our scheme initially reduces the number of generated clusters by generating large-sized clusters. In addition, our scheme minimizes the separation of the clusters even if the clusters are under an attack.

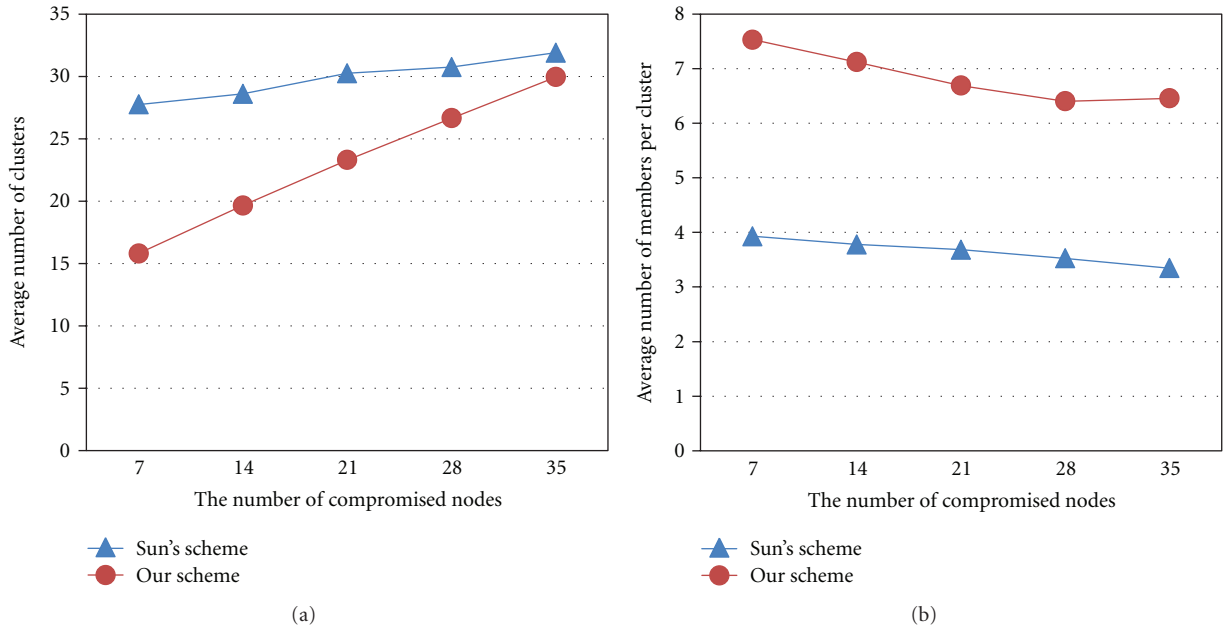

Figure 8: Variation in the number of clusters and the number of members per cluster (a) Avg. number of clusters versus compromised nodes (b) Avg. number of members per cluster versus compromised nodes.

Table 2: Simulation parameters.

Parameter	Value
Simulation area	100 m. × 100 m.
Number of nodes	100
Number of compromised nodes	7~35
Initial energy	20 Joules
Energy consumption model	Energy model of [1]
Bandwidth	1 Mbps
Packet header size	25 bytes
Transmission range	25 meters
Signature algorithm	ECDSA (elliptic curve digital signature algorithm)-160
Data encryption and decryption algorithm	AES (advanced encryption standard)-128
Hash Algorithm	SHA-1

Figure 8(b) shows how two schemes affect the average size of clusters. The result shows the quality of clusters when the clusters are under attacks. As shown in Figure 8, both schemes deteriorate the quality of clusters as the number of compromised nodes increases. Sun's scheme excludes a node whenever the node is identified as a compromised node or a suspicious node. So, it gradually deteriorates the quality of clusters as the number of compromised nodes. Our scheme keeps the quality of clusters higher than Sun's scheme because it separates much fewer nodes from clusters compared to Sun's scheme.

When a normal node identifies that a node has deviated from the protocol, it is likely to exclude the suspicious node from the cluster. As a result, many single clusters and double clusters are generated during the cluster formation. A single cluster consists of only one node, and a double cluster consists of only two nodes. Note that the clusters in Figure 8(a) do not include single clusters. Single and double clusters are almost meaningless in the viewpoint of clustering, and we label them as bad clusters. Figures 9(a) and 9(b) show how many bad clusters two schemes generate. As shown in Figures 9(a) and 9(b), Sun's scheme generates more bad clusters than our scheme. Especially, because Sun's scheme generates more single clusters than our scheme, we can say that it generates more useless clusters in comparison to our scheme.

Figure 10(a) shows how well two schemes remove compromised nodes from clusters during the protocol. The survived attackers in Figure 10(a) signify the nodes which have been compromised during the cluster formation and have survived until the end of the protocol. The result represents the healthiness of generated clusters in two schemes. Sun's scheme outperforms our scheme under a small number of compromised nodes (i.e., 7). Sun's scheme expels compromised nodes well through its one-hop conformity check when their population is small. However, because compromised nodes are not going to respond to normal nodes' protocol conformity check, the increase of compromised nodes significantly deteriorates performance. Whenever a node finds any inconsistency in the cluster membership, it attempts to obtain proofs from neighboring witnesses. However, if all witnesses are compromised, the node can never obtain any proof. So, the compromised node which causes the inconsistency can survive the protocol conformity check.

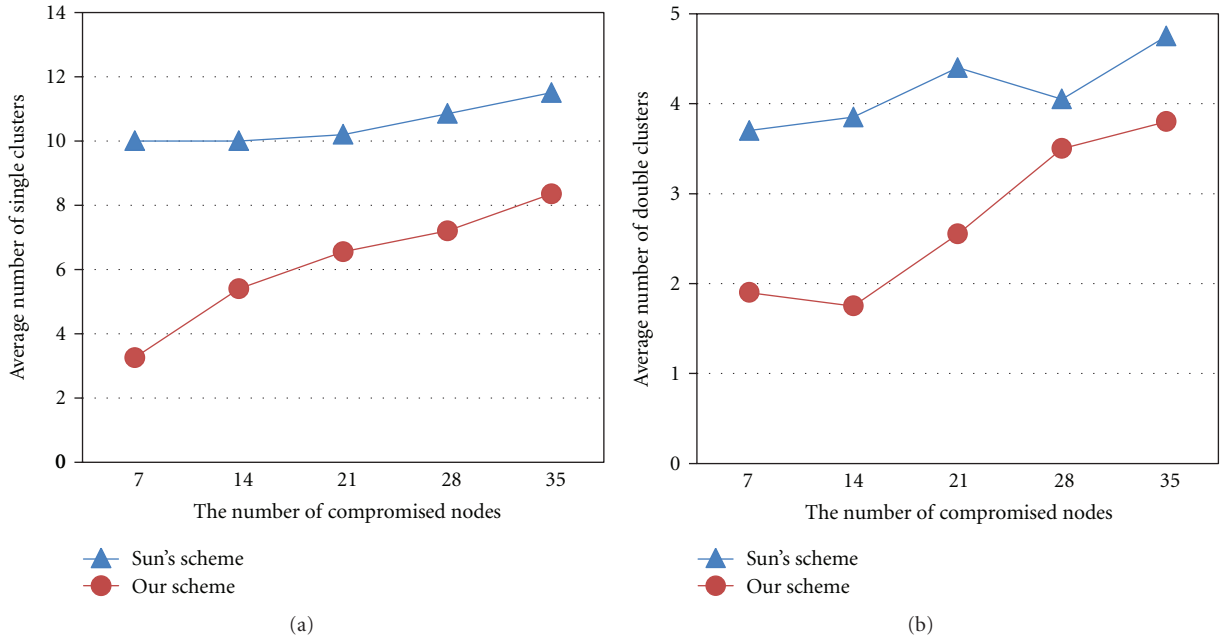

FIGURE 9: Variation in the number of bad clusters (a) Avg. number of single clusters versus compromised nodes (b) Avg. number of double clusters versus compromised nodes.

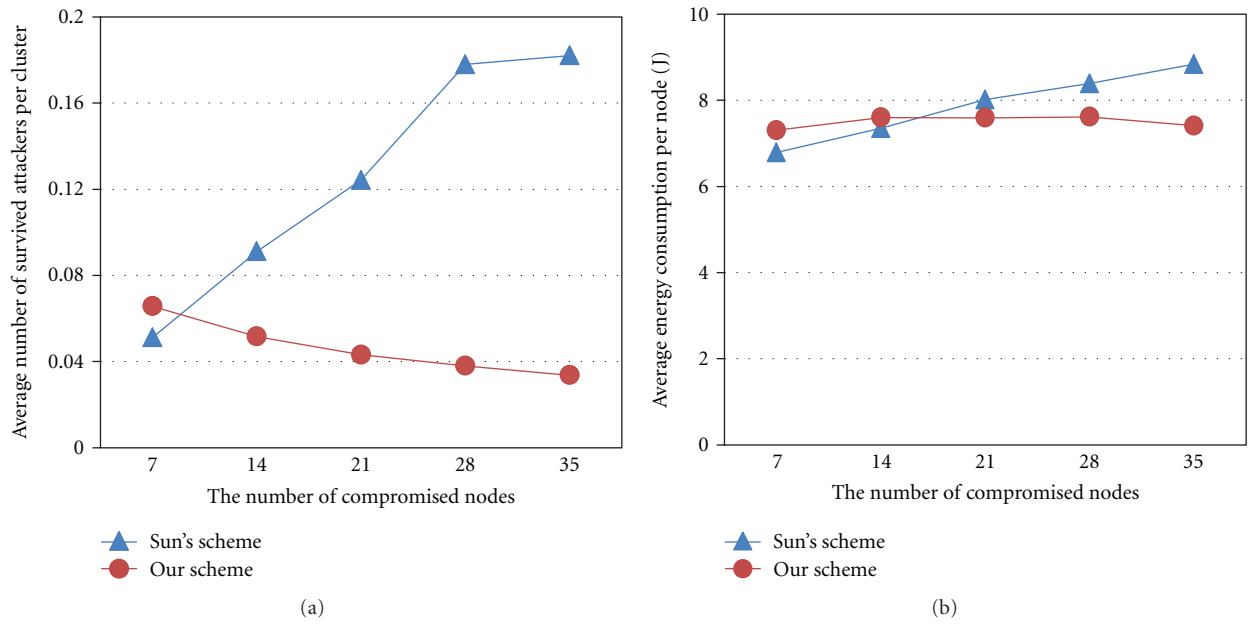

FIGURE 10: Variation in the number of survived attackers and energy consumption per node (a) Avg. number of survived attackers per cluster versus compromised nodes (b) Avg. energy consumption per node versus compromised nodes.

That is, the survival rate of compromised nodes increases when the population of compromised nodes grows up. However, our scheme employs two-hop transmissions to check the protocol conformity of a node. So, our scheme can obtain more proofs than Sun's scheme, and it can expel the compromised nodes regardless of the unresponsive nodes. Especially, as the number of compromised nodes increases, our scheme removes more compromised nodes from the network using the two-hop transmissions as shown in Figure 10(a).

Figure 10(b) shows how the increase of compromised nodes affects the energy consumption at each node. In Sun's scheme, if a compromised node causes any inconsistency of cluster membership between any two nodes, a normal node checks the protocol conformity of the node with the inconsistent membership. During the protocol conformity check

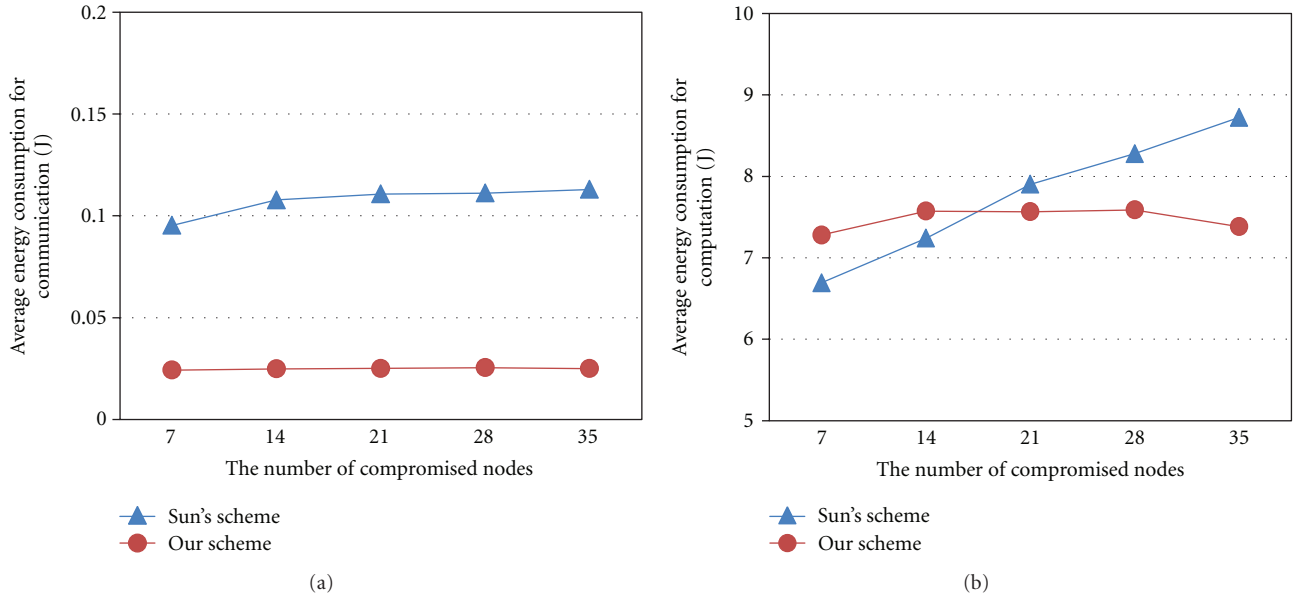

Figure 11: Variation in energy consumption for communication and computation (a) Avg. energy consumption for communication versus compromised nodes (b) Avg. energy consumption for computation versus compromised nodes.

process, the normal node requests the neighboring members to provide the messages previously received with a signature in a unicast manner. So, if compromised nodes increase, more normal nodes find the inconsistency, and they all begin the protocol conformity check of neighboring members. Consequently, the neighboring members consume much more energy than less compromised nodes. Even though our scheme increases the energy consumption of nodes under a small number of compromised nodes, it is a transient increase. As the number of compromised nodes increases, our scheme greatly reduces energy consumption of nodes as shown in Figure 10(b). This is because our scheme suppresses the employment of unicast transmissions and mainly employs broadcast transmissions.

Figure 11(a) shows the amount of energy each node consumed for communication during the protocol. As shown in Figure 11(a), Sun's scheme slightly increases energy consumption as the number of compromised nodes increases. Our scheme greatly decreases energy consumption as well as preserves the amount of energy consumption constantly regardless of the population of compromised nodes Figure 11(b) shows the amount of energy each node consumed for computation during the protocol. Both schemes consumed much more energy for computation than that for communication. It shows that the total energy consumption of both schemes highly depends on the energy consumption for the computation of both schemes. As shown in Figure 11(b), Sun's scheme incrementally increases the amount of energy consumption as the number of compromised nodes increases. This is mainly caused by the increase of one-hop conformity checks. Our scheme preserves the amount of energy consumption almost constantly regardless of the increase of compromised nodes. As a result, it greatly reduces the energy consumption of nodes especially under many compromised nodes.

Now, we compare the storage overhead of two schemes. Let Ni be the number of neighbors of node i. In Sun's scheme, because each node i should store the messages of all neighbors in each step, and the messages are received from the neighbors during the four steps, its storage overhead becomes $4Ni$ messages. If node i detects that a neighbor j has inconsistent cluster membership, it should receive an extra message from j and store it to complete the verification. In this case, node i's storage overhead is $4Ni + 1$ messages. Let M be the number of members in a cluster. So, we have an inequality of $4Ni > M(\approx 2Ni) > Ni$ according to the result of Figure 8(b). In our scheme, each node first stores Ni messages which are sent from its neighbors due to the exchange of certificates. Besides, each node i should store all CR messages sent from its members in order to build a consensus on the cluster membership. So, each node's storage overhead is $M + Ni$ messages. When a normal node does not receive any FCS message, it requests other members to send a proof of its normalcy. If the node succeeds in obtaining the proof, it should also store the proof message to persuade other members of the CS node's misbehavior. In this case, the node's storage overhead is $M + Ni + 1$ messages. According to the above analysis, our scheme's storage overhead is lower than Sun's scheme.

6. Conclusions

In this paper, we proposed a cluster formation scheme, which generates large-sized clusters in a secure manner. Our

scheme initially produces large sized clusters and preserves them using the two-hop conformity verification. Then, our scheme merges the cluster formation initiators into clusters only when they conform to the cluster formation protocol. Security analysis proves that our scheme can easily identify compromised nodes which cause any inconsistency in cluster membership and preserves the agreement in cluster membership against the compromised nodes. Simulation results show that our scheme reduces the number of generated clusters, and it suppresses the generation of useless clusters. Other simulation results prove that our scheme enhances the quality of clusters and expels more compromised nodes than the compared scheme. The simulation for energy efficiency evaluation proves that our scheme greatly reduces the amount of energy consumption, especially under many compromised nodes.

Acknowledgment

This work was supported by research funds of Chonbuk National University in 2011.

References

[1] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[2] O. Younis and S. Fahmy, "HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004.

[3] G. Gupta and M. Younis, "Performance evaluation of load-balanced clustering of wireless sensor networks," in *Proceedings of the 10th International Conference on Telecommunications (ICT '03)*, vol. 2, pp. 1577–1583, March 2003.

[4] G. Wang, K. S. Song, and G. Cho, "DIRECT: dynamic key renewal using secure cluster head election in wireless sensor networks," *IEICE Transactions on Information and Systems*, vol. E93-D, no. 6, pp. 1560–1571, 2010.

[5] G. Wang and G. Cho, "Clustering-based key renewals for wireless sensor networks," *IEICE Transactions on Communications*, vol. E92-B, no. 2, pp. 612–615, 2009.

[6] M. Sirivianos, D. Westhoff, F. Armknecht, and J. Girao, "Non-manipulable aggregator node election protocols for wireless sensor networks," in *Proceedings of the 5th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt '07)*, pp. 1–10, April 2007.

[7] Q. Dong and D. Liu, "Resilient cluster leader election for wireless sensor networks," in *Proceedings of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '09)*, pp. 108–116, June 2009.

[8] G. Wang and G. Cho, "Secure cluster head election using mark based exclusion inwireless sensor networks," *IEICE Transactions on Communications*, vol. E93-B, no. 11, pp. 2925–2935, 2010.

[9] A. C. Ferreira, M. A. Vilaca, L. B. Oliveira, E. Habib, H. C. Wong, and A. A. Loureiro, "On the security of cluster-based communication protocols for wireless sensor networks," in *Proceedings of the 4th IEEE International Conference on on Networking*, pp. 449–458, 2005.

[10] L. B. Oliveira, H. C. Wong, M. Bern, R. Dahab, and A. A. F. Loureiro, "SecLEACH-A random key distribution solution for securing clustered sensor networks," in *Proceedings of the 5th IEEE International Symposium on Network Computing and Applications (NCA '06)*, pp. 145–152, July 2006.

[11] D. Liu, "Resilient cluster formation for sensor networks," in *Proceedings of the 27th International Conference on on Distributed Computing Systems (ICDCS '07)*, pp. 40–48, 2007.

[12] K. Sun, P. Peng, P. Ning, and C. Wang, "Secure distributed cluster formation in wireless sensor networks," in *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC '06)*, pp. 131–140, December 2006.

[13] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[14] H. Rifà-Pous and J. Herrera-Joancomartí, "A fair and secure cluster formation process for Ad hoc networks," *Wireless Personal Communications*, vol. 56, no. 3, pp. 625–636, 2011.

[15] Y. C. Hu and A. Perrig, "Wormhole attacks in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 370–379, 2006.

[16] A. S. Wandert, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom '05)*, pp. 324–328, March 2005.

[17] D. R. Raymond and S. F. Midkiff, "Denial-of-service in wireless sensor networks: attacks and defenses," *IEEE Pervasive Computing*, vol. 7, no. 1, pp. 74–81, 2008.

[18] J. R. Douceur, "The sybil attack," in *Proceedings of the 1st International Workshop on Peer-to-peer Systems*, March 2002.

[19] B. Parno, A. Perrig, and V. Gligor, "Distributed detection of node replication attacks in sensor networks," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy (S&P '05)*, pp. 49–63, May 2005.