PAPER

SCODE: A Secure Coordination-Based Data Dissemination to Mobile Sinks in Sensor Networks

LeXuan HUNG[†a)], ***Nonmember***, **Sungyoung LEE**[†], **Young-Koo LEE**[†], ***and*** **Heejo LEE**[††], ***Members***

SUMMARY For many sensor network applications such as military, homeland security, it is necessary for users (sinks) to access sensor networks while they are moving. However, sink mobility brings new challenges to secure routing in large-scale sensor networks. Mobile sinks have to constantly propagate their current location to all nodes, and these nodes need to exchange messages with each other so that the sensor network can establish and maintain a secure multi-hop path between a source node and a mobile sink. This causes significant computation and communication overhead for sensor nodes. Previous studies on sink mobility have mainly focused on efficiency and effectiveness of data dissemination without security consideration. In this paper, we propose a secure and energy-efficient data dissemination protocol — *Secure COodination-based Data dissEmination* (SCODE) — for mobile sinks in sensor networks. We take advantages of coordination networks (grid structure) based on *Geographical Adaptive Fidelity* (GAF) protocol to construct a secure and efficient routing path between sources and sinks. Our security analysis demonstrates that the proposed protocol can defend against common attacks in sensor network routing such as replay attacks, selective forwarding attacks, sinkhole and wormhole, *Sybil* attacks, *HELLO* flood attacks. Our performance evaluation both in mathematical analysis and simulation shows that the SCODE significantly reduces communication overhead and energy consumption while the latency is similar compared with the existing routing protocols, and it always delivers more than 90 percentage of packets successfully.

key words: *sensor networks, sink mobility, routing, security, secure routing*

1. Introduction

A wireless sensor network (WSN) is composed of a large number of sensor nodes that are densely deployed either inside the phenomenon or very close to it. In sensor networks, a source is defined as a sensor node that detects a stimulus, which is a target or an event of interest, and generates data to report the stimulus. A sink is defined as a user that collects these data reports from the sensor network. Sinks may be mobile during its communication. In many sensor network applications such as military, homeland security, environment surveillance, it is necessary for sinks to access sensor networks while they are moving. For examples, in a battle field, a soldier with a PDA in hand continuously collects tank movement information while he is moving.

Sink mobility brings new challenges to secure routing in large-scale sensor networks. A mobile sink has to constantly propagate its current location to all sensor nodes so that the sensor network can establish and maintain a multi-hop path between the sink and a source. Doing this causes significantly communication overhead for the sensor network. On the other hand, nodes on the new path have to discover its neighbors and exchange secret information with each other to establish a secure communication. This also requires a lot of computation and communication overhead.

A number of routing protocols have been proposed for WSNs such as LEACH [1], SecRout [2], SEEM [3], SeRINS [4], TTSR [23]. However, these works do not take into account of sink mobility. Therefore, they are not suitable to work in such mobile sink environments because their routing paths are always static. Several studies have concerned the scalability and efficiency of data dissemination from multiple sources to multiple, mobile sinks such as TTDD [17], Directed Diffusion (DD) [18], SAFE [5], SEAD [6], Declarative Routing Protocol [7], and GRAB [8]. However, the authors were focusing only on energy efficiency without security consideration. Thus, these approaches are very vulnerable from many attacks in sensor network routing such as spoofed, altered, or replayed routing information [9], [16], selective forwarding, sinkhole [16], Sybil [10], wormhole [11], HELLO flood (unidirectional link) attacks [16].

In this paper, we present a secure and energy-efficient data dissemination protocol from multiple sources to multiple, mobile sinks for WSNs, namely Secure COordination-based Data dissEmination (SCODE). This work extends our previous study [12] with several improvements on the data dissemination scheme and an additional part — security. In the SCODE, the network plane is partitioned into a virtual grid with identical cells based on *Geographical Adaptive Fidelity* (GAF) [15] protocol. Each cell maintains one node, called coordinator, to participate in network communication while the other falls into *sleeping* mode. Each node stores three types of key [15] to establish a secure path from a source to a sink. The main contributions of this paper are two folds: (1) we propose a first, novel secure routing protocol specifically for mobile sinks in WSNs, and (2) our protocol achieves better energy efficiency than other popular approaches such as TTDD and DD while guaranteeing its average end-to-end delay comparable with TTDD and it successfully delivers most of the packets.

The rest of the paper is organized as follows. Section 2 briefly discusses on some existing routing protocols as well as their vulnerabilities and shortcomings. We then shortly describe GAF protocol in Sect. 3. The overview of our pro-

Manuscript received January 13, 2008.
Manuscript revised July 11, 2008.
[†]The authors are with the Dept. of Computer Engineering, Kyung Hee University, Korea.
[††]The author is with the Dept. of Computer Science and Engineering, Korea University, Korea.
a) E-mail: lxhung@oslab.khu.ac.kr
DOI: 10.1587/transcom.E92.B.131

posed protocol is presented in Sect. 4. In Sect. 5, we describe the SCODE in details. Section 6 presents the security analysis of our approach against common attacks in the sensor network routing. Section 7 shows performance evaluation in both mathematical analysis and simulation results. Finally, we conclude the paper in Sect. 8.

2. Related Work

Research on the sensor network routing has been carried out for nearly a decade. Heinzelman et al. [1] introduce a clustering algorithm for WSNs, called LEACH. In LEACH, sensors are organized into clusters. Each cluster has one *cluster head* (*CH*) which collects and aggregates information from its members (non-CH sensors in the same cluster) and passes on information to the *base station* (*BS*). However, LEACH has a number of shortcomings. LEACH assumes every node can directly reach the base station by transmitting with sufficiently high power. However, one-hop transmission directly to the base station is not feasible in large-scale WSNs due to the resource-limitations of sensors. On the other hand, LEACH is vulnerable from several attacks including *HELLO* flood, selective forwarding, and *Sybil* attacks [16].

Recent works have taken into account of both performance and security such as SecRout [2], SEEM [3], SeRINS [4], and TTSR [17]. SecRout [2] guarantees that sinks receive correct queries resulted from the sensor network. Only a high efficient symmetric cryptography is used in SecRout. SEEM [3] uses a principle similar to the Client/Server software architecture. The BS performs the route discovery and maintenance as well as route selection. Instead of a single path, the BS periodically selects a new path from multi-paths based on the current energy level of nodes along each path. SeRINS [4] focuses on detecting and isolating compromised nodes. The major shortcoming of those approaches is that they assume the BS is stationary and all sensor nodes know the BS's location. This assumption makes them fail to work in case the BS (sink) is mobile. TTSR [17] is a secure and efficient routing protocol for heterogeneous sensor networks (HSNs) which takes advantage of powerful high-end sensors (H-sensors) in an HSN. TTSR possesses a similar sink mobility problem with SecRout, SEEM, and SeRINS. On the other hand, TTSR is only suitable for HSNs with sufficient powerful sensor nodes, not large-scale homogeneous WSNs. Besides, relying only on some particular nodes makes them prone to deplete there energy sooner or later. Using fixed coordinators makes attackers easy to choose 'right' nodes to compromise.

There are a number of routing protocols aiming to support mobile sinks in WSNs [5]–[7], [17], [18]. Declarative Routing Protocol [7], and Directed Diffusion (DD) [18] suggest that each mobile sink needs to continuously propagate its location information throughout the sensor field, so that all sensor nodes get updated with the direction of sending future data reports. However, frequent location update from multiple sinks leads to both increased collisions in wireless transmissions and rapid power consumption of the sensor's limited battery supply. SAFE [5] uses flooding that is geographically limited to forward the query to nodes along the direction of the source. Considering the large number of nodes in WSNs, the network-wide flooding may introduce considerable traffic. SEAD [6] considers the distance and the packet traffic rate among nodes to create a near-optimal dissemination. SEAD strikes a balance between end-to-end delay and power consumption that favors power saving over delay minimization. It is therefore only suitable for applications with a less strict delay requirement. TTDD [17] exploits local flooding within a local cell of a virtual grid which sources build proactively. However, it does not optimize the path from the source to the sinks. When a source communicates with a sink, the restriction of grid structure may multiply the length of a straight-line path by . Also, TTDD frequently renews the entire path to the sinks. It therefore increases energy consumption and connection loss ratio. Those multi-hop routing protocols face many security problems from spoofed, altered, or replayed routing information [9], [16], selective forwarding [16], sinkhole [16], Sybil [10], wormhole [11], and *HELLO* flood (unidirectional link) attacks [16].

- Spoofing, altering, or replaying routing information [9], [16]: By spoofing, altering, or replaying routing information, the adversaries can create routing loops, attract or repel network traffic, extend or shorten source routes, generate false error messages, partition the network, increase end-to-end latency, etc.
- Selective forwarding attacks [16]: Malicious nodes may refuse to forward certain messages and simply drop them, ensuring that they are not propagated any further. Another form of this attack is an adversary selectively forwards packets, i.e. she is interested in suppressing or modifying packets originating from a few selected nodes, reliably forwards the remaining traffic and limits suspicion of her wrongdoing.
- Sinkhole and wormhole [16] attacks: In sinkhole attacks, the adversary attracts nearly all the traffic from a particular area through a compromised node, creating a metaphorical sinkhole with the adversary at the center. With wormhole attacks, adversary tunnels messages received in one part of the network over a low latency link and replays them in a different part. Wormhole attacks more commonly involve two distant malicious nodes colluding to understate their distance from each other by relaying packet along and out-of-bound channels available only to the attacker.
- Sybil attacks [10]: A single node presents multiple identities to other nodes in the network. In particular, a *Sybil* attack causes a significant threat to geographic routing protocols. Using a *Sybil* attack, an adversary can cheat as many nodes at different locations.
- HELLO flood (unidirectional link) attacks [16]: A laptop-class attacker may broadcast routing or other information with large enough transmission power and

convinces every node in the network that the adversary is its neighbor. As a consequence, these nodes only relay packages to the attacker's laptop.

3. Geographical Adaptive Fidelity (GAF) Protocol

SCODE is based on GAF protocol [15] to establish a coordination network. In GAF, a network plane is defined as a virtual grid, so that there only one node (called *coordinator*) in each cell stays awake to handle routing, while the others fall into *sleeping* mode.

GAF is based on an energy model that considers energy dissipation in sent/received packets and idle time, suggests that energy optimizations must turn off the radio, not simply reduce packet transmission and reception. In GAF, node location information and virtual grid are used to determine node equivalence. Two nodes are equivalent if they locate in the same virtual cells. The size of each virtual cell is determined based on the nominal radio range *R*. Assume the virtual grid is a square with *r* units on a side as show in Fig. 1. In order to meet the definition of virtual cell, the distance between two possible farthest nodes in any two adjacent cells, such as cell B and C in Fig. 1, must not be larger than *R*. For example, node 2 of cell B and node 5 of cell C in Fig. 1 are at the end of the long diagonal connecting two adjacent cells. Therefore, we get $r^2 + (2r)^2 \leq R^2$ or $r \leq R/\sqrt{5}$.

In GAF, nodes are in one of three states: *sleeping, discovery, active*. A state transition diagram is shown in Fig. 2. Initially nodes start out in the *discovery* state. When in state *discovery*, a node turns on its radio and exchanges discovery message to find other nodes within the same cell. The discovery message is a tuple of node id, cell id, estimated node active time (*enat*), and node state. As described above, a node uses its location and cell size to determine the cell id.

When a node enters *discovery* state, it sets a timer for T_d seconds. When the timer fires, the node broadcasts its

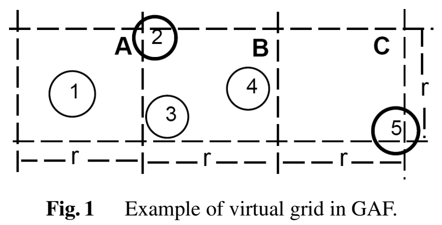

Fig. 1 Example of virtual grid in GAF.

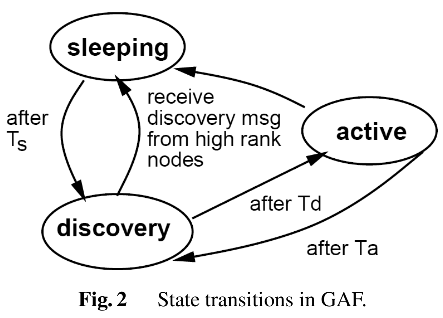

Fig. 2 State transitions in GAF.

discovery message and enters state *active*. The timer can also be suppressed by other discovery messages. This timer reduces the probability of discovery message collision.

When a node enters *active*, it sets a timeout value T_a to define how long this node can stay in *active* state. After T_a, the node will return to *discovery* state. While *active*, the node periodically re-broadcast its discovery message at interval T_d.

A node in *discovery* or *active* states can change state to *sleeping* when it can determine some other equivalent nodes will handle routing. When transitioning to *sleeping*, a node cancels all pending timers and powers down its radio. A node in the *sleeping* state wakes up after an application-dependent sleep time T_s and transitions back to discovery.

4. SCODE Overview

In this section, we present an overview of the SCODE. We first describe our assumptions as follows:

- There may be hundreds or thousands of homogeneous nodes uniformly distributed into a sensor field.
- All sensor nodes are stationary and aware of their locations. Sensors may use a secure location discovery service, e.g. [13], to estimate their locations, and no GPS receiver is required.
- Due to resource constraints, sensor nodes are not equipped with *tamper-resistant* hardware. If an adversary successfully compromises a sensor, then she can obtain all key material, data, and code stored on that node.
- Each node/sink has a unique id and maintains three key types based on LEAP scheme [14]: a unique individual key K_A that each node *A* shares with a sink; a cluster key K_{CH} shared among all nodes in the same cell; and a pairwise shared key K_{AB} shared between a node/sink *A* and its neighbor[†] *B*.
- Sinks are powerful nodes, moving within the sensor network field. Sinks are also aware of its location. Since sinks are mobile, sensors cannot know sink locations.
- Each sink stores a table containing (node, key) pairs (ID_{node}, K_{node}) of all the sensor nodes.
- Sinks are trusted.

Table 1 provides notation descriptions which will be used through out the paper.

In SCODE, the network plane is partitioned into a virtual grid with identical cells. Nodes in each cell negotiate based on GAF [15] so that there is only one node, called coordinator, stays awake to handle routing while the others fall into *sleeping* mode for the sake of energy. After an interval, sleeping nodes wake up and elect again. When a stimulus appears, sensors surrounding it collectively process signal and one of them becomes a source to generate data reports.

[†]Node *B* is defined as a neighbor of node *A* if and only if *B* coexists in the same or adjacent cell of *A*.

Table 1 Notations.

Notation	Description
ID_A	Identification of node A
$CID_A(X,Y)$	Cell identification of node A, indicating X and Y axes of the cell
PID	Packet Sequence Number (packet ID)
$A(x,y)$	Coordinate of node A
R	Radio range of a sensor node
r	Cell size
K_A	A secret key held by node A
K_{AB}	A shared key between A and B.
MAC (K, M)	Message authentication code of message M using a symmetric key K
$\{M\}$K	Message M is encrypted with a key K
N_0 , N_1	*Nonces*, one-time random number generated by nodes
$A \rightarrow$ broadcast: M	Node A broadcasts a message M
$A \rightarrow B$: M	Node A sends a message M to node B

SCODE has three major phases: *data announcement, query transfer* and *data dissemination*. In *data announcement* phase, the source broadcasts *data announcement* messages encrypted by its individual key to all coordinators. When a sink receives a *data announcement* message from a coordinator, it builds a query message, encrypted by the source's individual key and sends to the source. The query message is relayed through coordinators towards the source. During the *query transfer*, a routing path is established between the sink and the source. Upon receiving the query, the source starts generating data report and sends to the sink along the routing path. The sink frequently checks its current cell ID. If it moves out of the current cell, it first sends a *cache-removal* message to remove the old path. Then it sends a new query to the source to update its new location and establishes a new routing path. In order to detect a compromised or malfunctioning node, we introduce an *inspecting system*. In the *inspecting system*, all eight neighboring coordinators of each coordinator play roles of inspectors. The inspectors observe messages sent out from the inspected node and detect whether it is a compromised or malfunctioning node. For example, if a malicious node B receives a message from a node A and attempts to modify it before forwarding to C, the common inspectors of A and B will detect the change because they also receive the same messages sent out from A and B, so that they know the message originated from B is not the same as the original message sent out from A.

5. SCODE Algorithm

5.1 Establishment

5.1.1 Grid Formation

Supposed there are N sensor nodes deployed in the network area of a virtual grid. Each node itself computes cell ID $[X, Y]$ based on its coordinate (x, y) as follows:

$$X = \left\lfloor \frac{x}{r} \right\rfloor, \quad Y = \left\lfloor \frac{y}{r} \right\rfloor \tag{1}$$

In our scheme, we consider 9 surrounding cells as adjacent ones, so the total number of adjacent cells is 9, not 4 as GAF's. Therefore, the cell size r must be satisfied $(2r)^2 + (2r)^2 \leq R^2$ or $r \leq R/\sqrt{8}$ so that every node is connected with other nodes in adjacent cells.

5.1.2 Secure Neighbor Discovery

After deployment, each node A broadcasts a *HELLO* message in order to discover its neighborhoods. This message is encrypted by A's cluster key K_{CH}:

$$A \rightarrow broadcast\text{: } \{ID_A|CID_A|N_0\}K_{CH}$$

Each receiving node B decrypts the message and checks if it is a neighbor of A. If yes, it replies to A node ID and cell ID along with a *Nonce* value N_0 encrypted by the pairwise shared key K_{AB}.

$$B \rightarrow A\text{: } \{ID_B|\ CID_B|\ N_0{+}1\}K_{AB}$$

Receiving node A decrypts the message using the pairwise shared key. It then checks if the nonce N_0 is the one it has broadcasted. If it is, A accepts B as a neighbor and updates its neighborhood table.

The above two-way handshake protocol can avoid (or defend against) the unidirectional link problems (or attacks) [16]. For example, if an attacker uses node A which is a more powerful node such as a laptop with longer transmission range than B, then A can send a message to B directly, but B cannot send a message to A in one-hop. However, node B still thinks that A is a one-hop neighboring node, and various problems may arise, for example B will not relay messages to its neighbors but A, consequently these messages will be dropped.

5.2 Three Main Phases

5.2.1 Phase 1 — Secure Data Announcement Phase

When a stimulus is detected, a source, node S, propagates a *data-announcement* (DA) message to all coordinators using a *flooding* mechanism. The message contains source ID, cell ID and a *MAC*:

$$\begin{aligned}&S \rightarrow broadcast : PID|ID_S|CID_S|\{DA\}K_S|\\&MAC(K_S, PID|ID_S|CID_S|\{DA\}K_S)\end{aligned}$$

Every coordinator stores a few pieces of information for route discovery, including the information of the stimulus and the source location. In SCODE, source location does not mean the precise location of the source, but its cell ID. Since the coordinator role might be changed over time, the cell ID is the best solution for relaying the queries to the target nodes. In order to avoid indefinite storing *data-announcement* message in each coordinator, the source attaches a *timeout* parameter in each *data-announcement* message. Within the timeout interval, if the coordinator has not received any further *data-announcement* message, it removes the information of the stimulus and the source location to free the cache.

5.2.2 Phase 2 — Secure Query Transfer and Route Discovery

Receiving a *data-announcement* message, a sink looks up in its $\{(ID_{node}, K_{node})\}$ table, finds the key K_S shared with the source, and uses it to decrypt the message. It then constructs a query message using K_S and sends back to the source via coordinators as follows:

- The sink first sends to its agent (which is the closest coordinator):

  sink → *agent*:
  $PID \mid CID_{this} \mid CID_{next} \mid ID_S \mid CID_S \mid ID_{sink} \mid \{QUERY\}K_S \mid MAC\ (K_{sinkagent}, PID \mid CID_{this} \mid CID_{next}) \mid MAC\ (K_S, ID_S \mid CID_S \mid ID_{sink} \mid \{QUERY\}K_S)$

  where CID_{this} is cell ID of the agent, and CID_{next} is cell ID of the next cell (in this case $CID_{this} = CID_{sink}$, and $CID_{next} = CID_{agent}$). The source key K_S is used to encrypt the query content. It is also used to build a *MAC* of ID_S, CID_S, and ID_{sink} in order to provide data authentication and data integrity of the information sent from the sink. The pairwise shared key $K_{sinkagent}$ is used to build a *MAC* of *PID*, CID_{this}, and CID_{next} in order to provide data authentication and data integrity of the message sent out of the current node.
- The agent computes next cell ID towards the source, and then forwards the packet to the next cell:

  agent → *nextcell*:
  $PID \mid CID_{this} \mid CID_{next} \mid ID_S \mid CID_S \mid ID_{sink} \mid \{QUERY\}K_S \mid MAC\ (K_{agentnextcell}, PID \mid CID_{this} \mid CID_{next}) \mid MAC\ (K_S, ID_S \mid CID_S \mid ID_{sink} \mid \{QUERY\}K_S)$

  CID_{next} is computed using the algorithm illustrated in Fig. 3.
- Receiving coordinator checks whether its cell ID is the same as CID_{next} in the message. If yes, it computes next cell ID, and relays the message to the next cell. To reduce computation overhead of next cell ID calculation, each node maintains a routing table which maintains the departure cell ID, the destination cell ID, the uplink cell ID (which it receives the message) and the downlink cell ID (which is next cell ID CID_{next}).

The algorithm shown in Fig. 3 is described as follows:

- [Lines 4-7] The node first computes disparities ϕX, ϕY between source's cell and current node's cell. For example, cell ID of the current node is $CID_{this}(X,Y)$ = [3,1]; and cell ID of the source is $CID_S(X, Y)$ = [0,3]. Then:

```
1.  function next_cell_calculation()
2.  input: $CID_S$, $CID_{this}$
3.  output: $CID_{next}$
4.  $\Delta X = CID_S.X - CID_{this}.X$ ;
5.  $\Delta Y = CID_S.Y - CID_{this}.Y$;
6.  $\phi X = (\Delta X == 0)?0:\Delta X/|\Delta X|$;
7.  $\phi Y = (\Delta Y == 0)?0:\Delta Y/|\Delta Y|$;
8.  $CID_{next}.X = CID_{this}.X + \phi X$;
9.  $CID_{next}.Y = CID_{this}.Y + \phi Y$;
10. if (lookup_neighbor_table($CID_{next}$) == FALSE)
11. round_way_calculation();
12. }
```

Fig. 3 The algorithm to find next cell.

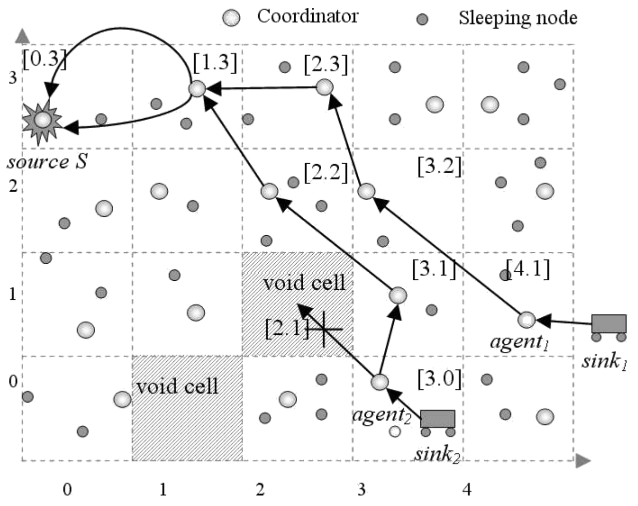

Fig. 4 Multi-hop routing through coordinators.

$$\Delta X = CID_S.X - CID_{this}.X = 0 - 3 = -3$$
$$\Delta Y = CID_S.Y - CID_{this}.Y = 3 - 1 = 2$$

So, the disparities are:

$$\phi X = (\Delta X == 0)?0 : \Delta X/|\Delta X| = -3/|3| = -1$$
$$\phi Y = (\Delta Y == 0)?0 : \Delta Y/|\Delta Y| = 2/|2| = 1.$$

- [Lines 8-9] Cell ID of the next hop is calculated as follows:

$$CID_{next}.X = CID_{this}.X + \phi X = 3 + (-1) = 2$$
$$CID_{next}.Y = CID_{this}.Y + \phi Y = 1 + 1 = 2.$$

  So CID_{next} = [2,2].
- [Line 10-11] The node then looks up in its neighborhood table. If there exists a coordinator in the next cell, it forwards the message to that cell; otherwise it considers the next cell a void cell. In this case, it finds a round-way. The round way calculation is simple: we can set the disparity either ϕX or ϕY equal to *zero* and re-calculate again the next cell ID in step 2.

As an example in Fig. 4, the $sink_1$ sends a query to the source along the route $\{sink_1$, [4,1], [3,2], [2,3], [1,3], $S\}$. However, in case of the $sink_2$, the coordinator in cell [3.0] can not find any node in cell [2.1] (due to void cell). Therefore, it finds a round way as {$sink_2$, [3,0], [3,1], [2,2], [1,3], S}. The sink re-sends queries if it moves to another cell.

Theorem 5.1. The *next_cell_calculation* algorithm guarantees that any message reaches its destination eventually.

Proof: Suppose the current node is not a destination ($ID_{this} \neq ID_S$ and $CID_{this} \neq CID_S$) and the node needs to compute the next cell ID. We prove that computed next cell is always towards to the destination.

Since $CID_{this} \neq CID_S$, we have at least $CID_{this}.X \neq CID_S.X$ or $CID_{this}.Y \neq CID_S.Y$. This means that at least $\phi X \neq 0$ or $\phi Y \neq 0$. Therefore, the computed next cell must locate between the current cell and the destination cell, i.e. the computed next cell is towards to the destination.

5.2.3 Phase 3 — Secure Data Dissemination

When a source receives a query from a sink, it starts generating reports and transmits to the sink. Data messages are encrypted by using the source key K_S. The source first sends to its uplink node A:

$S \rightarrow A$:
$PID \mid CID_{this} \mid CID_{next} \mid ID_S \mid CID_S \mid ID_{sink} \mid \{DATA\}\ K_S \mid MAC\ (K_{SA}, PID \mid CID_{this} \mid CID_{next}) \mid MAC\ (K_S, ID_S \mid CID_S \mid ID_{sink} \mid \{DATA\}\ K_S\ )$
where $CID_{this} = CID_S$, and $CID_{next} = CID_A$

Receiving node A checks if the data packet is forwarded to it or not by comparing its cell ID with CID_{next} in the packet. If not, it keeps the packet for a short time for inspecting purpose before dropping it (see Sect. 5.5). Otherwise, it computes next cell ID, changes PID, CID_{this},CID_{next} and compute a new MAC to replace MAC (K_{SA}, $PID \mid CID_{this} \mid CID_{next}$). After that, it relays the packet to the next cell.

When the message reaches the sink, the sink verifies MAC value in the message to ensure its data authentication and integrity. It then decrypts the message by using K_S.

5.3 Sink Mobility Management

Periodically, the sink checks the distance to the agent. If it recognizes that it no longer reaches the agent within the sensor transmission range, the sink has to compute new cell ID and selects a coordinator in that cell as a new agent. Cell ID is computed by the formula (1). Then, the sink re-sends a query to the source to establish a new data dissemination route by using the same mechanism described in Sect. 5.2.2. By re-sending queries only when the sink moves out of transmission range, SCODE significantly reduces communication overhead compared with other approaches. Hence, collision and energy consumption are reduced. Also, the number of loss data packet is decreased. In case the sink moves into a void grid, it selects the closest coordinator among adjacent cells as a new agent.

5.4 Coordinator Election

As aforementioned, SCODE is based on GAF [15] to establish a coordination network. During *discovery* phase, a node with the highest ranking will stay awake and play a role of a coordinator while the others fall into *sleeping* mode. The node ranking is determined by an application-dependent ranking procedure (it can be an arbitrary ordering of nodes to decide which nodes would be active, or it can be selected to optimize overall system lifetime). In GAF [15], a node with a longer expected lifetime is assigned a higher rank. This rule put nodes with longer expected lifetime into use first. However, this is very vulnerable. An adversary can use a compromised node to advertise the highest rank so that this node can be active all the time.

Practically, it is very hard (perhaps not possible) to absolutely defend against node compromise attacks for GAF. Therefore, we modify the *discovery* process and the ranking rule of GAF in order to reduce the risk without lessening its advantages. A coordinator is orderly selected according to node ID at each *discovery* round. For example, supposed there are m nodes $\{ID_1, ID_2 \ldots, ID_m\}$ ($ID_1 < ID_2 < \ldots < ID_m$) in a certain cell; if node ID_i is a coordinator at the current round, then ID_{i+1} would be a coordinator at the next round; if $i = m$ then $i + 1$ would be 1. By doing this way, the adversary cannot make her node active all the time at will. The compromised node is active only if it is a coordinator at the given round. On the other hand, our enhancement reduces communication overhead compared with GAF since nodes do not need to broadcast *discovery* messages.

5.5 Inspecting System

One of the most security concerns in sensor networks is node compromise. Due to resource limitation, sensor nodes are not equipped with any tamper-resistant hardware. Once a node is compromised, the adversary can extract all information stored in that node including all key materials. She then can use this node to perform various types of attacks to the networks. Defending against node compromise in routing is a non-trivial task.

In SCODE, we propose an *inspecting system* which utilizes neighboring coordinators to detect if a node performs any wrongdoing. Supposed A is a compromised node. So there are eight neighboring coordinators $\{B_1, B_2 \ldots\ B_8\}$ of A locating in eight adjacent cells. We call those *inspectors*. We also assume at least one node among those is legitimate.

Theorem 5.2. Given a destination cell ID and a current cell ID, there exists only one next hop cell ID and any node can compute by itself.

Proof: according to lines 4-9 of the algorithm in Fig. 3, CID_S (source/destination's cell ID) and CID_{this} (current cell ID) are unique, therefore CID_{next} must be unique. Thus, there exists only one next hop cell ID obtained from the algorithm. Since every node knows this algorithm, it can easily compute next cell ID if it knows CID_S and CID_{this}.

If a compromised node modifies routing information such as ID_S, CID_S, ID_{sink}, CID_{sink}, CID_{this} in the message, that can be easily detected by the uplink and downlink nodes because those nodes also maintain the source, sink, and information about its neighbors. In case the compromised node A modifies CID_{next} and then relays to a wrong cell, the inspectors can detect this by overhearing the message sent out of A and computing next cell ID based on the information in the message. If the actual next cell ID is not the same as next cell ID in the message, then the inspectors will report this as a problem. If A drops a packet, then the next cell node (uplink node) (which is also one of the inspectors) will detect by looking at packet sequence number. If A tries to modify the message content, it can be recognized by either next cell node or the source/sink. For example, if packet ID (*PID*) was modified, then the uplink node can detect that by verifying MAC using the pairwise shared key K_{SA}; if encrypted data $\{DATA\}K_S$ is modified, then the source/sink can detect by verifying the MAC with the source key K_S.

Once the compromised node is detected, the inspectors alert to other coordinators and eliminate that node from joining routing process. Coordinators consider that cell a void cell and establish another route by finding a round path.

6. Security Analysis

In this section, we analyze the security of SCODE in terms of sensor network routing attacks. As mentioned in Sect. 2, attacks on sensor network routing have been discussed in several papers [9]–[11], [16]. Most of the attacks fall into one of following categories: spoofed, altered, or replayed routing information [9], [16], selective forwarding [16], sinkhole [16], *Sybil* [10], wormhole [11], *HELLO* flood (unidirectional link) attacks [16]. We discuss how SCODE can defend against those.

- Spoofing, altering, or replaying routing information: In SCODE, this can be defended by using the *inspecting system* (Sect. 5.5).
- Selective forwarding attacks: SCODE prevents against selective forwarding attacks by using packet sequence numbers to uniquely identify each packet. If a compromised node selectively drops packets, that will be detected by uplink nodes.
- Sinkhole and wormhole attacks: In the algorithm *next_cell_calculation*, messages are always relayed towards the sink. Therefore, the adversary cannot attract traffic to a particular point, thus she can not launch sinkhole or wormhole attacks in SCODE.
- Sybil attacks: In *query transfer* and *data dissemination* phases, MAC is generated using the pairwise shared key of the sender and the receiver. In broadcasting process, a node uses the cluster key to encrypt the message and requires the receiver sends the reply encrypted by pairwise shared key. Therefore, a node cannot pretend to be another node. As a consequence, *Sybil* attack would not work.
- HELLO flood (unidirectional link) attacks: Nodes broadcast encrypted *HELLO* messages using cluster keys with *Nonce* values. That requires receiving nodes reply messages along with with received *Nonce* encrypted by the pairwise shared key. If an adversary is out-of-range of the sender, it cannot receive messages, thus cannot know the *Nonce* value. Therefore, SCODE can defend against the *HELLO* flood attacks.

7. Performance Evaluation

We evaluate SCODE performance and compare with existing approaches. The evaluation was carried out in both mathematical analysis and simulation. Firstly, we analyze communication overhead. Secondly, we present our simulation results including energy consumption, packet delivery ratio, and latency.

7.1 Communication Overhead Analysis

In this section, communication overhead is analyzed and compared with with TTDD [17] and other Sink-Oriented Data Dissemination approaches (henceforth called SODD) such as Directed Diffusion [18], GRAB [8]. Since query aggregation and data aggregation techniques are adopted in SCODE, TTDD, and SODD as well, we would not consider those in our overhead computation. We assume a similar system model and notations defined in [17]. Parameters are defined in Table 2. A summary of communication overhead analysis is shown in Table 3.

Supposed N nodes are uniformly deployed in a sensor field of A square meters. Each cell has a size $r \times r$ where $r = R/\sqrt{8}$ (R is the nominal radio range of sensor nodes). This cell size definition guarantees connectivity between two adjacent cells. There are k sinks moving with max speed v (m/s), while receiving d data packets from a source within a time period of $T(s)$. We assume that each non-security data packet (i.e. without security controls) is l_{data}(bytes) size long, and the other non-security messages (including broadcasting messages, queries, *cache-removal* messages, and *data-announcement* messages) are l_{query} (bytes) size long. As discussed in TinySec [19], the

Table 2 Parameter description.

Parameter	Description
A	Area of sensor field (m^2)
$c \times c$	Number of cells in virtual grid, $c = \lceil \sqrt{A}/(R\sqrt{8}) \rceil$
r	Cell size $r \le R/\sqrt{8}$
k	Number of mobile sinks
m	Number of cells a mobile sink traverses
d	Number of data packages a sink receives from a source
d/m	Number of data packets a sink receives between two consecutive location updates.
l_{query}	Size of query, cache removal, HELLO, data announcement message (bytes)
l_{data}	Size of data message (bytes)

Table 3 Communication overhead summary.

Overhead	Value (bytes)
Size of MAC in each package	4 bytes
Size of a secure data message	$l_{data} + 8$
Size of a secure query message	$l_{query} + 8$
Size of a secure data announcement message	$l_{query} + 4$
Overhead of broadcasting HELLO message	$N.l_{query}$
Overhead of data announcement message to reach all coordinators	$c^2.(l_{query} + 4)$
Overhead of sending secure queries and removal message from a source to *k* mobile sinks	$k.m.c(2l_{query} + 12)$
Overhead of delivering *d* data packets from a source to *k* mobile sinks	$k.c.d.l_{data}$
Total Overhead	$k.m.c.(2l_{query} + 12) + k.c.d.l_{data} + c^2.(l_{query} + 4) + N.l_{query}$

choice of 4-bytes MAC is not detrimental in the context of sensor networks. So we apply CBC-MAC to generate 4-bytes MACs for every message. As described in Sect. 5, each data message is inserted 2 MACs (i.e. 8 bytes). Thus, each data message would have a size of $l_{data} + 8$ bytes. Similarly, each query message will have a size of $l_{query} + 8$ bytes, and each *cache-removal* message and *data-announcement* message will be $l_{query} + 4$-byte long. Broadcasting messages have the same size as non-security broadcasting messages because encryption does not increase the size of encrypted information [19]. To model sink mobility, we assume each sink traverses m cells where $m \leq 1 + v.T/(R/\sqrt{8}$. Consequently, each sink has to send m *cache-removal* messages and m *queries*, and receives d/m data packets between two consecutive location updates.

Supposed there are $c \times c$ cells in the whole sensor field (where $c = \left\lceil \sqrt{A}/(R\sqrt{8}) \right\rceil$; $\lceil x \rceil$ is the smallest number larger than x). We analyze the communication overhead in the worst-case, i.e. the source and the sink are furthest away from each other.

For a query from a sink to reach a source, it traverses c cells throughout the sensor field, in other words, it traverses c hops. If a sink traverses m cells, it has to send m queries to the source. Overhead of sending m queries from k mobile sinks to a source is $k.m(c(l_{query} + 8) + c(l_{query} + 4)) = k.m.c(2l_{query} + 12)$, including sending *cache-removal* messages. Similarly, overhead of transmitting d data packets from a source to k sinks is $k.c.d.l_{data}$. Plus overhead $c^2.(l_{query} + 4)$ for *data-announcement* message to reach all coordinators using flooding mechanism and overhead of broadcasting *HELLO* message to find neighbors $N.l_{query}$, we have:

$$CO_{SCODE} = k.m.c.(2l_{query} + 12) + k.c.d.l_{data} + c^2.(l_{query} + 4) + N.l_{query}$$

In SCODE, there is no overhead for constructing the virtual grid, since each node is aware of its coordinate and computes cell ID on its own. Also, there is no overhead for *Coordinator Election* process because each node elects itself based on node ID. The *Inspecting System* also does not produce extra messages. This is because sensor communication is omni-directional, when a node sends a message to another node, its inspectors also overhear the message. Therefore, the total overhead of SCODE is:

$$CO_{SCODE} = k.m.c.(2l_{query} + 12) + k.c.d.l_{data} + c^2.(l_{query} + 4) + N.l_{query} \quad (2)$$

For TTDD, as analyzed in [17], the total overhead is

$$CO_{TTDD} = N.l + \frac{4N}{\sqrt{n_1}}l + k.m_1.n_1.lV + k.c_1(m_1.l + d(l_{data} + 8))\sqrt{2N} \quad (3)$$

And for the *SODD,* communication overhead is:

$$CO_{SODD} = k.m_1.N.l + k.c_1.d(l_{data} + 8)\sqrt{N} \quad (4)$$

where m_1 is the number of cells that a mobile sink traverses ($m_1 \leq 1 + v.T/\alpha$, where α is the TTDD's cell size), $n_1$is the number of nodes in each cell ($n_1 = N.\alpha^2/A$), and $c_1\sqrt{N}$ is the average number of sensor nodes along the straight-line path from the source to the sink ($0 < c \leq \sqrt{2}$).

For example, we assume the sensor field is an area of $A = 2000 \times 2000\,\text{m}^2$. The number of mobile sinks k is 4, moving with speed v= 10 (m/s). We suppose m and m_1 reach their maximum value, i.e. $m = 1 + v.T/(R/\sqrt{8})$ and $m_1 = 1 + v.T/\alpha$, where the nominal radio range $R = 250$ (m), $T = 200$ (seconds), and TTDD's cell size $\alpha = 200$ (m). Supposed $c_1 = 1$, l_{query}= 36 (bytes), l_{data}= 64 (bytes) and d= 100 data packets. We vary the number of nodes N from 0 to 10,000 in order to show the predominance of SCODE in node density.

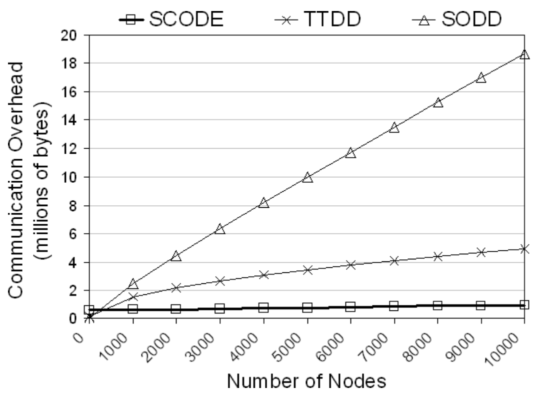

Fig. 5 Communication overhead vs. number of nodes.

Figure 5 plots the communication overhead of SCODE and a comparison with TTDD and SODD. The more the number of node is, the less overhead SCODE produces compared with TTDD and SODD. The reason is that only coordinators participate in sending and receiving packets in SCODE. Therefore, overhead only depends on the number of cells, instead of the number of nodes. Whereas, in TTDD and SODD, all nodes participate in communication, thus the total overhead increases as the number of nodes increases.

Table 4 Simulation parameters and defined values.

Simulation Parameters	Value
N (total nodes)	400 nodes
A (network size)	2000 x 2000 m^2
Transmission	0.66W
Reception	0.359W
Idle	0.035W
RC5 encryption of 64 bits	0.26 ms
Pseudorandom number generation	0.26 ms
4-byte MAC generation	0.13 ms
R	250 m
Data packet size	64 bytes
Other packet size	32 bytes
Number of mobile sinks	8
Speed of mobile sinks	10 m/s
Data generation interval	1 s

7.2 Simulation

7.2.1 Simulation Model

We simulated the SCODE on SENSE simulator (*Sensor Network Simulator and Emulator*) [20] and compared with TTDD [17] and DD [18]. The network comprises 400 nodes randomly deployed in a $2000 \times 2000\,\text{m}^2$ field. We use the same energy model used in *ns-2.1b8a* [21] that requires 0.66 W, 0.359 W and 0.035 W for transmitting, receiving and idling respectively. We set the power consumption rates of RC5 according to [22], [23] for encryption, MAC computation, and random number generation are 0.65 W, 0.48 W, and 0.36 W, respectively. As analyzed in [19], [24], we set the time consumption for encrypting 64 bits with RC5 0.26 ms, generating 64 pseudorandom bits takes 0.26 ms, and computing a 4-byte MAC requires 0.13 ms. The simulation uses MAC 802.11 *Distributed Coordination Function* (DCF) and nominal transmission range of each node is 250 m. *Two-ray ground* [25] is used as the radio propagation model. Each data packet has 64 bytes, query packets and the others are 36 bytes long. Additional bytes for MACs and nonce values are also put into each message. The default number of sinks is 8 moving with speed 10 m/s according to *random way-point model*. Two sources generate different packets at an average interval of 1 second. Summary of parameters and defined values are shown in Table 4.

7.2.2 Simulation Results

(a) Impact of Sink Number

As many users may simultaneously access the sensor network, it is important to consider the impact of the sink number. For the sensor network area of $2000 \times 2000\,\text{m}^2$, we set the number of sink varying from 1 to 8. Each sink moves with the maximum speed 10 m/s and a 5-second pause time. The number of total nodes and the number of sources are not changed. Figure 6 shows total energy consumption of SCODE as the number of sinks varies from 1 to 8. It demonstrates that SCODE is more energy efficient than TTDD and DD. This is because of two reasons. Firstly, SCODE relies on a coordination network, so that nodes in each cell negotiate among themselves to turn off its radio to significantly reduce energy consumption. Meanwhile, TTDD and DD must turn on all nodes to participate in routing. Secondly, SCODE optimizes a number of transmission hops between sources and sinks that is based on the cell size to maximize the communication distance between two adjacent cells. Figure 7 plots the average end-to-end delay of SCODE. The figure shows that the delay of SCODE is approximate to that of TTDD. In Fig. 8, it shows that the success rate of SCODE is always above 95 percent. It means that SCODE delivers most of packets successfully.

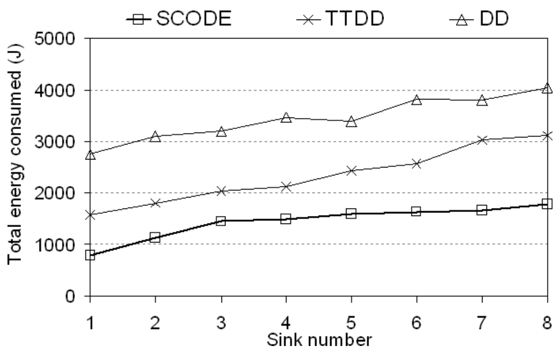

Fig. 6 Total energy vs. sink number.

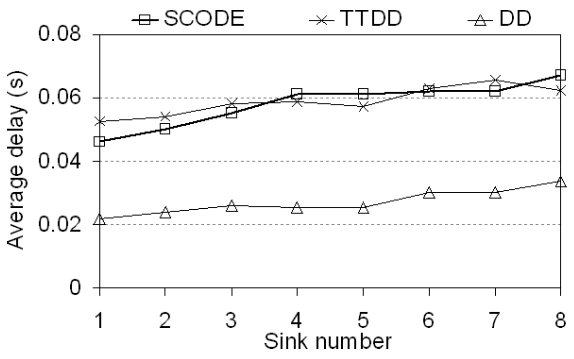

Fig. 7 Average delay vs. sink number.

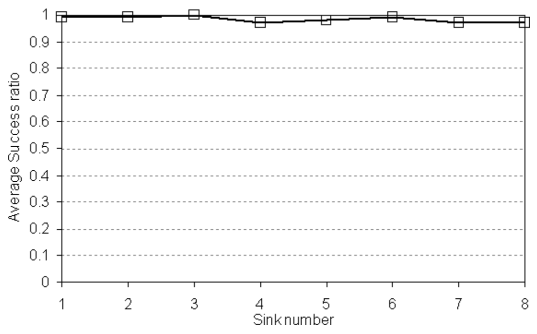

Fig. 8 Sucess ratio vs. sink number.

(b) Impact of Sink Mobility

In order to examine the impact of sink mobility, we

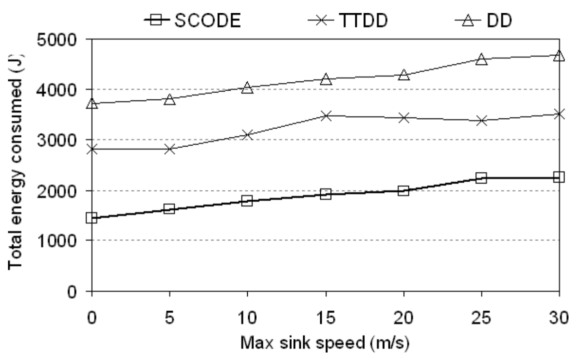

Fig. 9 Total energy vs. sink mobility.

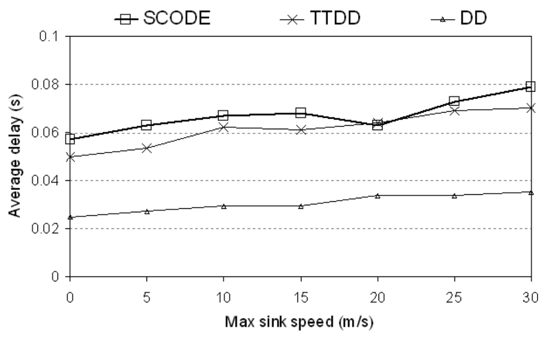

Fig. 10 Average delay vs. sink mobility.

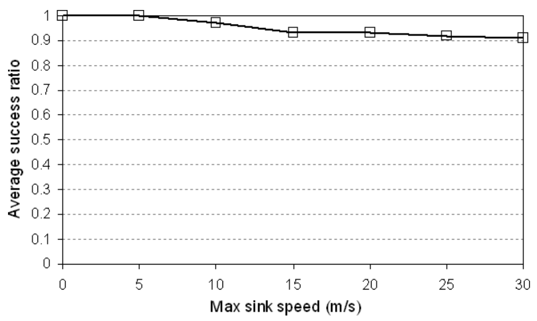

Fig. 11 Sucess ratio vs. sink mobility.

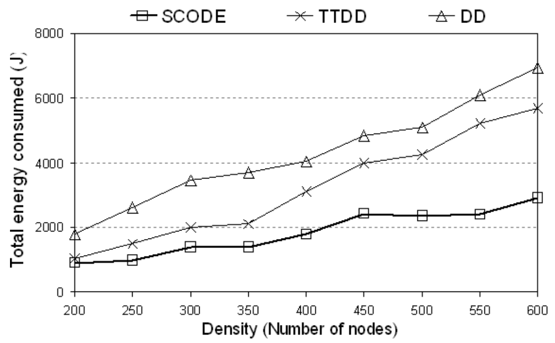

Fig. 12 Total energy vs. network density.

measure SCODE for different sink speeds (0 to 30 m/s). In this experiment, the network consists of 8 mobile sinks and 400 sensor nodes. The number of sources does not change. Figure 9 shows the total energy consumed as the sink speed changes. In both low and high speeds of the sinks, SCODE shows that the total energy consumed is much less than TTDD and DD. The reason is because, aside from above reasons, SCODE reduces the number of re-transmissions of query and updating sink's locations while the sinks are moving. The query only needs to resend as the sink moves to another cell. In contrast, TTDD and DD send more messages to propagate new location of the sinks throughout the sensor field to all sensor nodes. Figure 10 shows the delay of SCODE which is comparable with TTDD. In Fig. 11, as the sinks speed up, the average success ratio is always above 90 percent. This results show that SCODE handles mobile sinks efficiently.

(c) Impact of Node Density

To evaluate impact of node density on SCODE, we vary the number of nodes from 200 to 600. The number of sinks is 8. Each sink keeps moving with speed 10 m/s as the default setting. The number of sources is 2. The sensor field size is not changed. Figure 12 shows the energy consumption with different node densities. The figure demonstrates that SCODE consumes less energy than TTDD and DD. As the number of nodes increases, the total energy consumed slightly increases while that of TTDD and DD significantly increases. This is mainly because SCODE turns off radio most of the time. Therefore, energy is consumed mostly by the coordinators, while in TTDD and DD, nodes do not participate in communication still consume much energy in idling mode.

8. Conclusion

In this paper, we presented a new secure and energy-efficient routing protocol for mobile sinks in sensor networks, namely *Secure Coordination-based Data Dissemination* (SCODE) protocol. SCODE takes advantages of a coordination network based on GAF to apply a secure and efficient routing scheme. In SCODE, the sensor network plane is partitioned into a virtual grid with identical cells. The cell size is optimal to reduce the number of transmission hops while guaranteeing connectivity between every two adjacent cells. Sensor nodes in each cell negotiate among each other so that only one node, called coordinator, stays awake and the others fall into *sleeping* mode for the sake of energy. After a time interval, all nodes wake up and re-elect a new coordinator. By doing this, nodes do not quickly run out of its energy, thus prolonging the network lifetime. Communication between sources and sinks are established via coordinators and the shortest path is calculated. Keys in SCODE are generated by LEAP scheme. As the proof of concepts shown, SCODE can defend against several common attacks in the sensor network routing including spoofed routing information, selective forwarding, sink-hole, wormhole, Sybil, and *HELLO* flooding attacks. Both analysis and simulation results have shown that SCODE achieves good performance. By comparing with other popular protocols supporting multiple, mobile sinks: Two-Tier Data Dissemination (TTDD) and Directed Diffusion (DD), we have shown that SCODE is much more energy efficient while its average end-to-end delay is comparable with TTDD and the delivery ratio is always above 90 percent.

Acknowledgments

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Advancement) (IITA-2008-(C1090-0801-0002)). This research was supported by the MIC (Ministry of Information and Communication), Korea, Under the ITFSIP (IT Foreign Specialist Inviting Program) supervised by the IITA (Institute of Information Technology Advancement, C1012-0801-0003). This work is financially supported by the Ministry of Education and Human Resources Development (MOE), the Ministry of Commerce, Industry and Energy (MOCIE) and the Ministry of Labor (MOLAB) through the fostering project of the Lab of Excellency.

References

[1] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks,” IEEE Trans. Wireless Commun., vol.1, no.4, pp.660–670, Oct. 2002.

[2] J. Yin and S.K. Madria, “SecRout: A secure routing protocol for sensor networks,” 20th International Conference on Advanced Information Networking and Applications (AINA 2006), pp.393–398, April 2006.

[3] N. Nasser and Y. Chen, “SEEM: Secure and energy-efficient multipath routing protocol for wireless sensor networks,” Comput. Commun., vol.30, pp.2401–2412, 2007.

[4] S.-B. Lee and Y.-H. Choi, “A secure alternate path routing in sensor networks,” Comput. Commun., vol.30, pp.153–165, 2006.

[5] S.K. Son, S.H. Stankovic, J.A.S. Li, and Y. Choi, “SAFE: A data dissemination protocol for periodic updates in sensor networks,” Proc. 23rd IEEE International Conference on, pp.228–234, 2003.

[6] H.S. Kim, T.F. Abdelzaher, and W.H. Kwon, “Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks,” Proc. First ACM International Conference on Embedded Networked Sensor Systems, pp.193–204, Nov. 2003.

[7] D. Con, D.V. Hook, S. McGarry, and S. Kolek, “Declarative ad-hoc sensor networking,” Proc. SPIE Vol4126. Integrated Command Environments, Patricia Hamburger, ed., pp.109–120, Nov. 2000.

[8] F. Ye, G. Zhong, S. Lu, and L. Zhang, “GRAdient broadcast: A robust data delivery protocol for large scale sensor networks,” Wirel. Netw., vol.11, no.1, pp.285–298, May 2005.

[9] A.D. Wood and J.A. Stankovic, “Denial of service in sensor networks,” Computer, vol.35, no.10, pp.54–62, Oct. 2002.

[10] J.R. Douceur, “The Sybil attack,” Proc. IPTPS, pp.251–260, March 2002.

[11] Y.-C. Hu, A. Perrig, and D.B. Johnson, “Wormhole detection in wireless ad hoc networks,” Technical Report TR01-384, Department of Computer Science, Rice University, June 2002.

[12] L.X. Hung, D.H. Seo, S.Y. Lee, and Y.K. Lee, “Minimum-energy data dissemination in coordination-based sensor networks,” Proc. 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Application (RTCSA), pp.381–386, Hong Kong, Aug. 2005.

[13] L. Lazos and R. Poovendran, “SeRLoc: Secure range-independent localization for wireless sensor networks,” Proc. ACM Workshop Wireless Security, pp.21–30, Oct. 2004.

[14] S. Zhu, S. Setia, and S. Jajodia, “LEAP+: Efficient security mechanisms for large-scale distributed sensor networks,” ACM Trans. Sensor Networks, vol.2, no.4, pp.500–528, Nov. 2006.

[15] Y. Xu, J. Heidemannn, and D. Estrin, “Geography-informed energy conservation for ad hoc routing,” Proc. Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2001), pp.70–84, Italy, July 2001.

[16] C. Karlof and D. Wagner, “Secure routing in wireless sensor networks: Attacks and countermeasures,” First IEEE International Workshop on Sensor Network Protocols and Applications (SPNA), pp.113–127, May 2003.

[17] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, “TTDD: A two-tier data dissemination model for large-scale wireless sensor networks,” Wirel. Netw., vol.11, no.1-2, pp.161–175, Jan. 2005.

[18] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, “Directed diffusion for wireless sensor networking,” IEEE/ACM Trans. Netw., vol.11, no.1, pp.2–16, Feb. 2003.

[19] C. Karlof, N. Sastry, and D. Wagner, “TinySec: A link layer security architecture for wireless sensor networks,” Proc. 2nd International Conference on Embedded Networked Sensor Systems (SenSys04), pp.162–175, Nov. 2004.

[20] SENSE (Sensor Network Simulator and Emulator), http://www.cs.rpi.edu/˜cheng3/sense/

[21] NS-2, http://www.isi.edu/nsnam/ns/

[22] Q. Xue and A. Ganz, “Runtime security composition for sensor networks,” Proc. IEEE Veh. Technol. Conf., pp.105–111, Oct. 2003.

[23] D. Xiaojiang, G. Mohsen, X. Yang, and C. Hsiao-Hwa, “Two tier secure routing protocol for heterogeneous sensor networks,” IEEE Trans. Wireless Commun., vol.6, no.9, pp.3395–3401, Sept. 2007.

[24] Z. Benenson, F.C. Freiling, E. Hammerschmidt, S. Lucks, and L. Pimenidis, “Authenticated query flooding in sensor networks,” Proc. 4th Annual IEEE International Conference on Pervasive Computing and Communications Workshops, vol.201, pp.38–49, 2006.

[25] T.S. Rappaport, Wireless communications, principles and practice, Prentice Hall, 1996.

LeXuan Hung received the B.S. degree in Computer Science from Hanoi University, Vietnam, in 2003, and the M.S. degree in Computer Engineering from Kyung Hee University, Korea, in 2005. Currently, he is a Ph.D. candidate in Computer Engineering, Kyung Hee University. His research interests include sensor networks, network security, and ubiquitous computing.

Sungyoung Lee received his B.S. in Material Science from Korea University in 1978, M.S., Ph.D. in Computer Science from Illinois Institute of Technology, USA in 1987, and 1991, respectively. Since 1993, he has been a professor at the Dept. of Computer Engineering, College of Electronics and Information, Kyung Hee University, Korea. From 1992 to 1993, he was a assistant professor at the Dept. of Computer Science, Governors State University, University Park, USA. His research interest includes Ubiquitous Computing Middleware, Java Virtual Machine, Real-Time System, and Embedded System.

Young-Koo Lee received his B.S., M.S., Ph.D. in Computer Science from Korea Advanced Institute of Science and Technology (KAIST), Korea in 1988, 1994, and 2002, respectively. Since 2004, he has been an assistant professor at the Dept. of Computer Engineering, College of Electronics and Information, Kyung Hee University, Korea. From 2002 to 2004, he was a Post Doctoral Fellow Advanced Information Technology Research Center (AITrc), KAIST, Korea, and a Postdoctoral Research Associate at Dept. of Computer Science, University of Illinois at Urbana-Champaign, USA. His research interest Ubiquitous Data Management, Data Mining, Activity Recognition, Bioinformatics, On-line Analytical Processing, Data Warehousing, Database Systems, Spatial Databases, Access Methods.

Heejo Lee received his B.S., M.S., Ph.D. in Computer Science and Engineering from Pohang University of Science and Technology (POSTECH), Korea in 1993, 1995 and 2000, respectively. Since 2004, he has been an assistant professor at the Department of Computer Science and Engineering, Korea University, Korea. From 2001 to 2003, he was at Ahnlab, Inc. as Chief Technology Officer (CTO) and Director of Technology Planning Department. From 2000 to 2001, he was a Post Doctoral Research Associate at the Network Systems Lab of the Department of Computer Sciences, and at the Center for Education and Research in Information Assurance and Security (CERIAS), Purdue University. His research interest includes computer and communication security, parallel scientific computing, and fault-tolerant computing.