Efficient Geographical Information-Based En-route Filtering Scheme in Wireless Sensor Networks

Chuanjun Yi[1,2,3*], Geng Yang[1,3], Hua Dai[1,3], Liang Liu[4] and Yunhua Chen[5]

[1] College of Computer Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing 210003 - China
[2] Zijin College, Nanjing University of Science and Technology, Nanjing 210023 - China
[3] Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, Nanjing 210023 - China
[4] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016 - China
[5] Nanjing Institute of Mechatronic Technology, Nanjing 211135 - China
*Corresponding author: Chuanjun Yi

Received August 29, 2017; revised December 16, 2017; revised March 2, 2018; accepted April 17, 2018; published September 30, 2018

Abstract

The existing en-route filtering schemes only consider some simple false data injection attacks, which results in lower safety performance. In this paper, we propose an efficient geographical information-based en-route filtering scheme (EGEFS), in which each forwarding node verifies not only the message authentication codes (MACs), but also the report identifier and the legitimacy and authenticity of locations carried in a data report. Thus, EGEFS can defend against not only the simple false data injection attacks and the replay attack, but also the collusion attack with forged locations proposed in this paper. In addition, we propose a new method for electing the center-of-stimulus (CoS) node, which can ensure that only one detecting node will be elected as the CoS node to generate one data report for an event. The simulation results show that, compared to the existing en-route filtering schemes, EGEFS has higher safety performance, because it can resist more types of false data injection attacks, and it also has higher filtering efficiency and lower energy expenditure.

Keywords: Wireless sensor network, geographical information, false data injection, en-route filtering, compromised node

1. Introduction

Due to their advantages on various aspects such as sensitive information awareness, real-time data feedback, and flexible network organization, wireless sensor networks (WSNs) have a bright future and development potential in the fields of military applications, healthcare [1], industry [2], agriculture [3], urban monitoring [4], etc. However, the sensor nodes may be deployed in an unattended hostile environment, thus facing a high risk of being captured and compromised. Once a node is compromised, the adversary will get all the information stored in that node. As a result, the adversary can easily control it to launch false data injection attacks, i.e., to send bogus data reports to the data collection point (i.e., the sink), which will lead to not only false alarms but also the depletion of limited energy in a battery powered network. Although several recent researches [5-9] have proposed mechanisms for message authentication and node authentication in sensor networks, these solutions can only prevent the false data reports injected by outside attackers, but not those injected by compromised nodes (i.e., inner attackers).

To filter out the false data reports injected by compromised nodes, many schemes have been proposed [10-24]. In [10-13], the false data reports are forwarded to the sink by intermediate nodes without any filtering, and are verified when they reach the sink, which will waste the constrained resources of intermediate nodes and the limited network bandwidth. However, in the en-route filtering schemes proposed in [14-24], the false data reports are verified by intermediate nodes and are dropped early, which can save the limited network bandwidth and energy. Therefore, the false data en-route filtering strategies are favored by scholars. However, the existing en-route filtering strategies only consider some simple false data injection attacks, which results in lower security. How to design an efficient en-route filtering scheme to defend against more complicated attacks with less costs remains a challenging open problem.

In this paper, we design a complicated collusion attack, and propose an efficient en-route filtering scheme to defend against various types of false data injection attacks. The major contributions of this paper are as follows:

1) The existing CoS (Center-of-Stimulus) election method cannot ensure that only one CoS node will be elected for an event on all occasions, and there may be multiple reports of an event to be delivered to the sink, which will incur high communication overhead. To solve this problem, we propose an effective method for electing the CoS node, which ensures that only one detecting node will be elected as the CoS node to generate one data report for an event.

2) We propose a strategy based on report identifier, which enables the forwarding nodes to filter out duplicated data reports.

3) We design a new type of false data injection attack called the “collusion attack with forged locations”, in which the malicious compromised node forges the locations of the collaborative compromised nodes around the location of the fabricated event.

4) We propose an efficient geographical information-based en-route filtering scheme (EGEFS) and a sink verification scheme, which can defend against various types of false data injection attacks. Both theoretical analysis and simulation results demonstrate that EGEFS outperforms SEF [19], GFFS [23], and DSF [24] in terms of higher security, filtering efficiency, and energy saving.

The rest of the paper is organized as follows. Section 2 presents related works. The system models and assumptions are introduced in Section 3. Section 4 presents the detailed design of EGEFS and sink verification. The performance of algorithms is analyzed in Section 5 and the

simulation results are presented in Section 6. Finally, conclusion of this paper is drawn in Section 7.

2. Related Work

En-route filtering is an effective mechanism to address the false data injection problem, with which, the forwarding nodes can detect and drop false reports early. Recently, many en-route filtering schemes [14-24] have been proposed which can be classified into two major categories: symmetric cryptography based schemes and asymmetric cryptography based schemes. Asymmetric cryptography based schemes (e.g., CCEF [14], LTE [15], PDF [16], DAEF [17], and ERF [18]) are generally more resilient to adversaries, however, symmetric cryptography based schemes are more efficient in terms of communication overhead, computation overhead, and storage overhead. Therefore, symmetric cryptography based techniques have attracted huge attention, and large amount of work has been done in recent time, such as SEF [19], DEFS [20], GRPEF [21], BECAN [22], GFFS [23], NFFS [23], and DSF [24].

SEF [19] is a pre-deployment scheme in which each node randomly picks some secret keys from one partition of a global key pool before deployment. When an event occurs, each detecting node generates a MAC (Message Authentication Code). The CoS node randomly chooses T (T>1) MACs generated by using the keys of different partitions and attaches them to the event report. As a report is forwarded, each forwarding node verifies the correctness of the MACs probabilistically and drops the report with any invalid MAC. SEF is independent with data dissemination protocols. However, it has limited filtering probability and can only tolerate a small number of compromised nodes.

Yu et al. [20] presented a dynamic en-route filtering scheme called “DEFS”. In DEFS, the cluster head uses the Hill Climbing algorithm to disseminate the authentication keys of sensing nodes along multiple paths toward the sink, which guarantees that the forwarding nodes closer to a cluster hold more authentication keys than those far from it. In the filtering phase, each forwarding node verifies the reports and drops the false ones. DEFS can lighten the overhead of nodes close to the sink during reports forwarding. However, due to periodic maintenance of node association and key dissemination along multiple paths toward the sink, DEFS has low resilience to the number of compromised nodes, and incurs great energy cost.

Li et al. [21] proposed a scheme, referred to as grouping-enhanced resilient probabilistic en-route filtering (GRPEF). In GRPEF, an efficient distributed algorithm is proposed to divide sensor nodes into exact T groups, and guarantee that any location in the monitored area is covered simultaneously by T nodes from distinct groups with a high probability. Furthermore, a novel location-aware key derivation technique based on multiaxis division is proposed to tackle the threshold limitation of SEF. GRPEF is independent of sink stationary and routing protocols, and it can significantly improve the filtering effectiveness. However, the key derivation technique based on multiaxis division incurs high communication overhead.

Lu et al. [22] proposed a novel bandwidth-efficient cooperative authentication (BECAN) scheme to filter false data reports. Based on the characteristics of random graph and the cooperative bit-compressed authentication technique, BECAN can detect and filter the majority of injected false data reports early at the forwarding nodes, which can largely reduce the sink’s burden. With the multi-reports technology, BECAN can offer en-route filtering with a high probability, and it also has high reliability, but at the expense of high storage cost. Furthermore, the timestamp is embedded in each data report, which enables BECAN to defend against the replay attack. However, the timestamp technology requires high-precision clock

synchronization, and it is difficult to choose the size of time window. The SPINS scheme proposed in [13] resists the replay attack by using the counter (CTR) mode and nonce, which can bypass the problems encountered in the timestamp technology. However, SPINS cannot be applied directly by forwarding nodes to filter duplicated data.

To address the problem that the existing en-route filtering schemes cannot defend against collaborative false data injection attacks, Wang et al. [23] proposed a geographical information based false data filtering scheme (GFFS). In GFFS, the keys of sensor nodes are bound to their geographical locations, and each node distributes its location information to some other nodes after deployment. Each forwarding node verifies not only the correctness of MACs, but also the legitimacy of the locations. Thus, the false reports injected collaboratively by compromised nodes from different geographical areas can be detected and filtered out. However, distributing location information in the pre-deployment phase incurs longer latency and higher energy cost.

Wang et al. [23] further proposed a neighbor information based false data filtering scheme (NFFS). In NFFS, each node distributes its neighbor information to the forwarding nodes along its forwarding path to the sink after deployment. When a forwarding node receives a report, it verifies the correctness of the MACs and the legitimacy of relative positions of the endorsing nodes (the detecting nodes which endorse the report). Thus, false reports can be detected by checking the relationship between the keys of sensor nodes and their locations. NFFS requires no positioning devices, which reduces the cost. However, the storage cost in NFFS is higher than that in GFFS.

Sun et al. [24] proposed a double key-sharing based false data filtering scheme (DSF). In DSF, the sensor nodes are grouped into clusters. Each forwarding node picks an in-cluster node as its associated node, and shares a pair-wise key (called A-type key) with its associated node. Thus, each node in DSF stores k R-type keys and one A-type key. A legitimate report must carry two types of MACs (R-type MACs and A-type MACs). After receiving a report, a forwarding node validates both the two types of MACs. To reduce packet size, DSF uses the technique of Bloom filter in SEF.

We can see that the aforementioned schemes only consider some simple false data injection attacks launched by single compromised node (e.g., SEF, DEFS, GRPEF, and BECAN) or simple collusion attacks (e.g., GFFS, NFFS, and DSF). Besides, the existing CoS election method may elect multiple CoS nodes, so multiple reports of an event may be transmitted to the sink, which will incur high communication overhead.

In this paper, we present a complicated collusion attack, called "collusion attack with forged locations", which can evade the filtering of the existing en-route filtering schemes. We also propose an efficient geographical information-based en-route filtering scheme (EGEFS) which can resist multiple types of false data injection attacks, including the proposed complicated collusion attack. In addition, we propose an effective CoS election method which can ensure that only one CoS node will be elected to generate a data report for an event.

3. System Models and Assumptions

3.1 Sensor Network Model

We consider a wireless sensor network with N_a sensor nodes $v_1,\cdots,v_{N_a}$ and a sink node S, randomly placed within a square area of $Q\times Q$. We model the sensing range and communication range of a sensor as a circle centered at its actual location, with the radius R_s and R_c, respectively. Once deployed, the sink and all the sensor nodes are assumed to be static,

and each node can obtain its own location by GPS or other localization schemes [25-27].

We further assume that the sensor nodes are deployed in high density, so that an event can be detected by multiple nodes. Each detecting node produces a keyed MAC by using one of its stored keys. The CoS node collects the MACs, then randomly chooses T (T>1) MACs generated by using the keys of different partitions, and attaches them to the data report which is then forwarded to the sink through multi-hops.

3.2 Attack Models

In this paper, we assume the sink will never be compromised, and the sensor nodes may be compromised or physically captured. Once compromised, a node can be controlled by the adversary to inject false reports into the sensor network. The existing en-route filtering schemes mainly consider the following three attack models (i.e., Attack Model 1, Attack Model 2, and Attack Model 3).

1) Attack Model 1 [19-22]: attack by single compromised node

Each malicious compromised node, i.e., a compromised node which will launch attacks, only stores its own security information, so it needs to forge information (e.g., IDs, locations, key indices, keys, and MACs) for other T-1 endorsing nodes of different partitions to generate forged reports in legitimate forms. These forged reports are then forwarded to the sink.

2) Attack Model 2 [23]: collaborative false data injection attack

Wang et al. [23] considered a type of false data injection attack called “collaborative false data injection attack”. The adversary compromises multiple nodes from different geographical areas, and stores the information (e.g., IDs, locations, key indices, and keys) of other T-1 compromised nodes of different partitions into each malicious compromised node. Thus, each malicious compromised node can use such information to forge data reports.

3) Attack Model 3 [22]: replay attack

After compromising some nodes which have forwarded legitimate reports, the adversary can abuse them to inject the former reports into the network.

In the above attack models, the attacks in Attack Model 2 can pass the defense of some existing en-route filtering schemes (e.g., SEF and DSF), but cannot pass the defense of GFFS since each forwarding node verifies the legitimacy of the locations in the report (i.e., the location of each endorsing node should be within the sensing range of the event). However, if the adversary forges the locations of the endorsing nodes around the location of the fabricated event, then the forged report can pass the legitimacy verification of the locations in GFFS. Therefore, we propose a new type of false data injection attack, called “collusion attack with forged locations”, as shown below.

4) Attack Model 4: collusion attack with forged locations

Each malicious compromised node has the information of IDs, locations, key indices, and keys of at least T compromised nodes of different partitions. When forging a data report, the malicious compromised node forges the locations of the T collaborative endorsing nodes around the location of fabricated event, which enables the forged report to pass the legitimacy verification of locations in GFFS.

This paper aims to work out an efficient en-route filtering scheme which can resist the four aforementioned attacks to achieve higher security, while also considering the filtering efficiency, storage cost, and energy expenditure.

4. The Proposed Scheme

When designing our scheme, we need to address the following problems:

- How to assign the keys?
- How to generate a data report for an event?
- How to route a date report to the sink?
- How to filter a false report at a forwarding node?
- How to verify a data report at the sink?

Among which, the first and third problems will be addressed by using the existing methods, which will be introduced in Section 4.1, and the rest three problems will be addressed by using the proposed scheme. Next, we will introduce the solutions to these problems in detail.

4.1 Pre-deployment and Initialization

There are two tasks in this phase: 1) Before deployment, each node preloads some keys. 2) After deployment, the nodes are organized into a data collection tree rooted at the sink for routing event reports to the sink.

1) Key Assignment

We adopt the key assignment method in [19]. There is a pregenerated global key pool of N keys, denoted as $G=\{K_i|0\leq i\leq N\text{-}1\}$, which is divided into n ($n>T$) non-overlapping partitions $U=\{U_i|0\leq i\leq n\text{-}1\}$. Each partition contains m keys ($N=n\times m$), and each key has a unique key index. The way to partition the global key pool is as follows: $U_i=\{K_j|im\leq j\leq (i+1)m\text{-}1\}$. Before deployment, each node randomly picks one partition and stores k ($k<m$) keys from this partition, together with the associated key indices. The sink has complete knowledge of the global key pool and the secret information of all nodes.

2) Construction of Routing Tree

After deployment, the nodes are organized into a routing tree rooted at the sink for distributing and collecting information in the network. Every event report will be routed to the sink along the routing tree. We construct the routing tree by using the DSAPS scheme in [28], which enhances the network lifetime by reducing the load of the node with highest load. The routing tree can be updated periodically as needed.

After the routing tree is constructed, each node transmits its location to the sink along the routing tree, and all its neighbors store the location when they hear it. Finally, each node stores the locations of all its neighbors.

4.2 Report Generation

In this phase, we need to address two problems: 1) How to select the CoS node from the detecting nodes of the event? 2) How does the CoS node generate the event report?

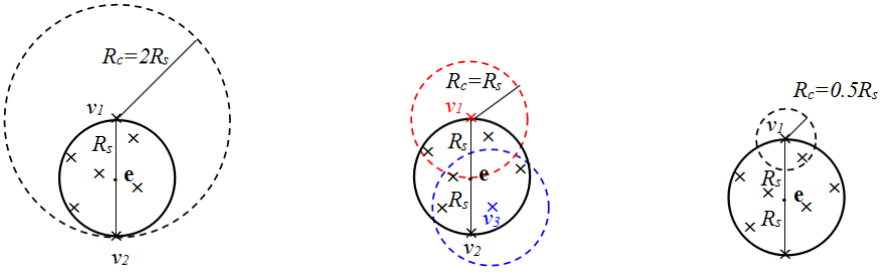
1) Election of CoS

The method of CoS election in SEF and GFFS is as follows: Each detecting node of the event sets a random timer and broadcasts its observed values of $\{L_E, t, E\}$ when the timer expires, where L_E is the location of the event, t is the time of detection, and E is the reading of the event. If another node finds its observed values are consistent with the broadcast values within the predefined error range, it accepts them and cancels its timer. Otherwise, it broadcasts its own observed values when its timer expires. The node whose broadcast values are accepted by the other detecting nodes becomes the CoS node.

When all the detecting nodes are neighbors of each other, the above method can ensure that only one detecting node will be elected as the CoS node for the same event. Otherwise, there may be multiple nodes elected as CoS nodes. Our analysis is as follows.

When $R_c \geq 2R_s$, all the detecting nodes of an event are neighbors of each other. This is because the maximum distance between any two detecting nodes is $2R_s$, and if $R_c \geq 2R_s$, then any detecting node is within the communication ranges of others, i.e., all the detecting nodes are neighbors of each other. For instance, just as shown in **Fig. 1(a)**, when $R_c=2R_s$, the circle drawn in solid line denotes the sensing range of the event e, and the circle drawn in dashed line denotes the communication range of the node v_1. We can see that v_2 is the farthest detecting node from v_1 with a distance of $2R_s$, yet v_2 is still within the communication range of v_1, i.e., v_2 is a neighbor of v_1. Obviously, all the other detecting nodes are neighbors of v_1. Similarly, we can infer that all the detecting nodes are neighbors of each other. Suppose that the scheduled time of v_1 is the shortest among all the detecting nodes of event e, then v_1 broadcasts its observed values of event e first, and all the other detecting nodes will receive the broadcast values and cancel their timers. Thus, v_1 becomes the only CoS node of event e.

When $R_c<2R_s$, the detecting nodes of an event are not always neighbors of each other. For instance, as shown in **Fig. 1(b)**, when $R_c=R_s$, the two circles drawn in dashed lines denote the communication ranges of v_1 and v_3 respectively. We can see that some detecting nodes (e.g., v_2 and v_3) are outside of the communication range of v_1, which means they are not neighbors of v_1. Suppose that v_1 broadcasts its observed values of event e first, and all its neighboring detecting nodes which received the broadcast values cancel their timers. Then, suppose v_3 broadcasts its observed values of event e, and all its neighboring detecting nodes cancel their timers. Finally, both v_1 and v_3 become CoS nodes of event e. The similar situation occurs in **Fig. 1(c)**, which will not be discussed in detail due to lack of space.



(a) $R_c=2R_s$ (b) $R_c=R_s$ (c) $R_c=0.5R_s$

Fig. 1. The influence of R_c and R_s on the neighbor relationship among detecting nodes

To ensure that only one detecting node will be elected as the CoS node to generate one data report for an event, we propose the following effective method for electing the CoS node.

When an event occurs, each detecting node of the event sets a timer, indicating that the CoS election phase begins. The timer can be set to $T_{\text{CoS}} = \left\lceil \frac{2R_s}{R_c} \right\rceil * T_{tran} * k_l$, where T_{tran} denotes the time used to transmit and receive a message between two neighboring nodes in wireless communication, and k_l is a coefficient which can be set according to the link quality and the delay requirements in specific application. Besides, each detecting node sets a random timer, denoted as t_1, and broadcasts its observed values of $\{ ID, C, L_E, t, E, t_1\}$ in a SC message when its timer expires, where ID is the identity of the detecting node, C is the value of the counter on the detecting node set for defending against the replay attack (which will be discussed in Section 4.3), L_E is the location of the event, t is the time of detection, and E is the reading of the event. When a detecting node v_j receives a SC message, it first detects whether the event in the SC message is the same as the event that it has observed. If not the same event, it drops the received SC message directly. Otherwise, v_j detects whether any SC message of the event has

been stored locally. If it has not stored any SC message of the event, v_j cancels the timer, stores the received SC message, and then broadcasts it. Otherwise, v_j compares t_1 in the received SC message with that stored locally. If the t_1 in the received SC message is smaller (i.e., the received SC message is an earlier message), then v_j overwrites the SC message stored locally with the received SC message, and then broadcasts it. Otherwise, v_j drops the received SC message. The CoS election phase ends when the timer T_{CoS} on each detecting node expires. The pseudo-code for election of CoS is shown in **Algorithm 1**. Each detecting node will know who is the CoS node from the *ID* in the SC message stored locally. If a detecting node finds itself to be the CoS node, it will start to prepare a data report for the event.

Algorithm 1: Election of CoS on Each Detecting Node (e.g., v_i)

```
v_i sets a timer denoted as T_cos;
v_i sets a timer denoted as t_1;
while (1) {
  if (v_i receives a SC message) {
    if (the event in the received SC message is the same event that v_i has observed) {
      if (any SC message of the event has been stored locally) {
        if (t_1 in the received SC message is smaller than that in the SC message stored
            locally)
          v_i overwrites the SC message stored locally with the received SC message, and
          then broadcasts it;
        else v_i drops the received SC message;
      }
      else
        v_i cancels the timer t_1, stores the received SC message, and then broadcasts it;
    }
    else v_i drops the received SC message;
  }
  if (v_i has not canceled the timer t_1 && the timer t_1 expires)
    v_i stores and broadcasts its observed values of { ID, C, L_E, t, E, t_1} in a SC message;
  if (the timer T_cos expires) break; //The CoS election phase ends.
}
```

2) Report Generation

After selecting the CoS node, each detecting node v_i randomly selects one key K_i from its k keys and generates a MAC:

$$M_i = \overline{\mathrm{MAC}}(K_i, ID_{CoS} \,||\, C \,||\, L_E \,||\, E) \tag{1}$$

where, || denotes stream concatenation, ID_{CoS} is the identity of CoS, and C is the value of the counter on the CoS. Formula (1) computes the MAC M_i of message $\{ID_{CoS}||C||L_E||E\}$ by using key K_i. Like SEF, we also use RC5 algorithm [29] as it involves only addition, XOR, and bit shifting operations, and it also has low memory requirement.

Next, each detecting node sends $\{ID, L, i, M_i\}$ to the CoS, where *ID* is its identity, L is its location, i is the key index of key K_i, and M_i is the keyed MAC using K_i. The CoS randomly chooses T detecting nodes of different key partitions (including itself) as the endorsing nodes, and attaches their $\{ID, L, i, M_i\}$ to the event report. The event report generated by the CoS looks like $\{ L_E, t, E, C, ID_1, L_1, i_1, M_{i1}, \ldots\ldots, ID_T, L_T, i_T, M_{iT} \}$, where ID_1 is ID_{CoS}. To reduce packet size, we use the technique of Bloom filter in [19] to map the MACs into a string of d bits: $F=b_0\, b_1 \ldots b_{d-1}$, using k_h independent hash functions $h_1, h_2, \ldots, h_{kh}$. Thus, the final report becomes $\{ L_E, t, E, C, ID_1, L_1, i_1, \ldots\ldots, ID_T, L_T, i_T, F\}$. Then, the CoS sends the report to the sink along the routing tree described in Section 4.1.

4.3 En-route Filtering

As reports are forwarded toward the sink, each forwarding node verifies the reports and drops the false ones, which can conserve energy and bandwidth resources of nodes along the forwarding paths.

To defend against the attacks in Attack Model 1, each forwarding node verifies the correctness of the MACs in the report probabilistically, as in SEF. That is, if the forwarding node has any of the keys in the report, it computes the corresponding MAC, then computes each hash value of the MAC and sees if the corresponding bit is "1" in F. The report is dropped if at least one of them is "0".

To resist the attacks in Attack Model 2, each forwarding node verifies the legitimacy of the locations in the report, as in GFFS. That is, the location of each endorsing node should be within the sensing range of the event. The report is dropped if there is any illegitimate location in the report.

For defense against the replay attacks in Attack Model 3, BECAN [22] adopts the timestamp technology. However, the timestamp technology requires high-precision clock synchronization, and it is difficult to choose the size of time window. If the time window is set small, the real event reports may be discarded by the forwarding nodes as duplicated reports. If the time window is set large, the duplicated reports can still be forwarded in the network for a while, which will result in a weak defense. Therefore, we propose a method based on report identifier, which enables the forwarding nodes to filter out duplicated reports in time. The details are described as follows.

Each sensor node is equipped with a counter whose value C is initialized to zero. Whenever the CoS node generates an event report, its counter value C is increased by one. When a forwarding node receives a report, it extracts the report identifier $\{ID_1, C\}$ from the report and compares it with the report identifiers stored locally. If the report identifier has been stored locally, then the report is considered to be duplicated, and it will be discarded directly. Otherwise, the report is considered to be fresh, and its identifier $\{ID_1, C\}$ will be stored locally. The concrete steps of the method are shown in **Algorithm 2**.

Algorithm 2: Detection of Duplicated Reports

1 Extract the report identifier $\{ID_1, C\}$ from the received report R.
2 Look for $\{ID_1, C\}$ in the local list of report identifiers which is initially empty.
3 If find the same record, then set *flag*=1, otherwise set *flag*=0.
4 If *flag*=1, then R is considered to be duplicated, return 0; Otherwise, R is considered to be fresh, store $\{ID_1, C\}$ into the local list of report identifiers, then return 1.

In order to resist the collusion attacks in Attack Model 4, we propose a strategy to verify the authenticity of locations in the report, which is performed by the one-hop forwarding node of the report. Suppose that v_i is the one-hop forwarding node of the report. If an endorsing node in the report is a neighbor of v_i, then v_i can directly check the authenticity of its location, since v_i has stored the locations of all its neighbors. Therefore, v_i only needs to verify the location authenticity of the endorsing nodes which are not its neighbors with the following steps.

Suppose that v_{IDj} is the current endorsing node to be verified, then v_i performs the GPSR algorithm [30] to send an ASK message to v_{IDj}, embedded with i (the identity of v_i), ID_j, and L_j. If v_{IDj} exists at location L_j, it will receive the ASK message, and send back a REPLY message embedded with i and ID_j to v_i along the same path in which it received the ASK message. Otherwise, v_{IDj} may not receive the ASK message and cannot send back the REPLY message to v_i. After sending an ASK message, v_i sets a timer. The timer can be set according to the

maximum distance between v_i and the endorsing nodes to be verified, the link quality of the WSN, and the delay requirements of specific application. If v_i has not received all the REPLY messages when the timers expire, it will drop the report. The pseudo-code for verifying the authenticity of locations in the report is shown in **Algorithm 3**.

By combining the above methods for defending against the four types of attacks, we can obtain EGEFS, as shown in **Algorithm 4**.

Algorithm 3: Authenticity Verification of Locations in the Report on Node v_i

```
flg=0;
for (1≤j≤T) {
    if (v_IDj is a neighbor of v_i) {
       if (L_j in the report is not the same as the location of v_IDj stored locally)
            return 1; // The verification fails.
    }
    else {
            flg=1;
            v_i executes the GPSR algorithm {
                 Find next hop node v_k;
                 Send an ASK message to v_k;
                 v_k and each successive next hop node execute the GPSR algorithm until the
                  ASK message reaches the location L_j;
            }
            v_i sets a timer;
            Wait for the REPLY message;
    }//end if
}//end for
if (flg=0 || v_i has received all the REPLY messages when the timers expire)
      return 0; // The verification succeeds.
else  return 1; // The verification fails.
```

Algorithm 4: En-Route Filtering on Node v_i in EGEFS

/* On receiving report R*/

1 Check that the format of R is as complete as { L_E, t, C, E, ID_1, L_1, i_1, ……, ID_T, L_T, i_T, F}, and there are at most $k_h \times T$ "1"s in F; drop R otherwise.

2 Check that the T key indices belong to T distinct partitions; drop R otherwise.

3 Perform **Algorithm 2** to check whether R is a duplicated report. Drop R if it is.

4 Check that ($| L_E, L_j | \le R_s$, $1 \le j \le T$); drop R otherwise.

5 If v_i has one key $K \in \{K_{i_j}, 1 \le j \le T\}$, it computes $M = \overline{\mathrm{MAC}}(K, ID_{CoS} \| C \| L_E \| E)$, then computes each hash value of M and see if the corresponding bit is "1" in F; drop R otherwise.

6 If v_i is the one-hop forwarding node of R, it performs **Algorithm 3** to verify the authenticity of locations in R. Drop R if the verification fails.

7 Send R to the next hop.

4.4 Sink Verification

The sink has all the keys, and knows the keys and locations of all the nodes. Therefore, even if some bogus reports escape en-route filtering and reach the sink, the sink can further verify and reject the false reports by following the steps in **Algorithm 5**.

Algorithm 5: Sink Verification

/* On receiving report R*/

1 Check that the format of R is as complete as { L_E, t, C, E, ID_1, L_1, i_1, ……, ID_T, L_T, i_T, F}, and there are at most $k_h \times T$ "1"s in F; drop R otherwise.

2 Check that the T key indices belong to T distinct partitions; drop R otherwise.

3 Perform **Algorithm 2** to check whether R is a duplicated report. Drop R if it is.

4 Check that ({ ID_j, L_j, i_j},$1 \le j \le T$) are the same as the corresponding information stored locally; drop R otherwise.

5 Check that ($| L_E, L_j | \le R_s$, $1 \le j \le T$); drop R otherwise.

6 For each key $K \in \{K_{i_j}, 1 \le j \le T\}$, compute $M = \overline{\mathrm{MAC}}(K, ID_{CoS} \| C \| L_E \| E)$, then regenerate the Bloom filter F' and compare with F. Drop R if $F' \neq F$.

5. Performance Analysis

In this section, we will analyze the security performance, energy expenditure, and storage overhead of EGEFS, and compare the performance of EGEFS with that of SEF, DSF, and GFFS.

5.1 Security Performance

SEF, DSF, GFFS, and EGEFS can defend against the attacks in Attack Model 1 to a certain extent. In SEF, the forwarding nodes can filter out bogus reports probabilistically by verifying the correctness of the MACs. In DSF, the forwarding nodes verify two types of MACs (R-type MACs and A-type MACs). Therefore, the filtering ability of DSF is stronger than that of SEF. In GFFS, the forwarding nodes verify not only the MACs, but also the legitimacy of the locations in the reports. Therefore, the filtering ability of GFFS is stronger than that of SEF and DSF. In EGEFS, the forwarding nodes verify not only the MACs and report identifiers, but also the legitimacy and authenticity of the locations in the reports. Therefore, the filtering ability of EGEFS is stronger than that of SEF and DSF, and no weaker than that of GFFS.

SEF and DSF cannot defend against the attacks in Attack Model 2. This is because the MACs in a bogus report are forged by using the keys of T compromised nodes of different partitions, which can pass the MAC authentication in SEF and DSF. GFFS and EGEFS can well defend against such attacks by verifying the legitimacy of the locations carried in the reports, and can filter out the false reports at the one-hop forwarding nodes.

SEF, DSF, and GFFS cannot defend against the replay attacks in Attack Model 3. In comparison, EGEFS can filter out the duplicated reports by verifying the report identifiers.

SEF and DSF cannot defend against the attacks in Attack Model 4, because the MACs in a bogus report are forged by using the keys of T compromised nodes of different partitions, which can pass the MAC authentication in SEF and DSF. In GFFS, each node distributes its location information to some other nodes after deployment. Thus, the forwarding nodes can detect the forged locations in the reports probabilistically and drop those with false locations. The number of location packets that each node distributes, c, affects the en-route filtering probability and energy expenditure. The larger the c, the higher the en-route filtering probability, but the higher the energy expenditure for distributing location information, and the higher the storage cost. EGEFS can filter out a bogus report within one hop by verifying the authenticity of locations in it. Therefore, EGEFS has stronger filtering ability than GFFS.

In conclusion, the security performance of SEF, DSF, GFFS, and EGEFS is shown in **Table 1**, in which "×" means that there is no filtering ability, "*" means that there is filtering ability, and more "*" indicates stronger filtering ability.

Table 1. Security performance

	Attack Model 1	Attack Model 2	Attack Model 3	Attack Model 4
SEF	*	×	×	×
DSF	**	×	×	×
GFFS	***	***	×	*
EGEFS	***	***	***	***

5.2 Energy Expenditure

Here, we overlook the energy expenditure for sensing and computing. Furthermore, SEF, GFFS, and EGEFS can use the same method of CoS election, and the energy expenditures for electing CoS are the same for all the three algorithms, which therefore can be ignored. Thus, the energy expenditure of SEF takes into account the communication overhead for forwarding the reports. For DSF, only the communication overhead for forwarding the reports is taken into account. The energy expenditure of GFFS takes into account two sources: One is the communication overhead for distributing location information during the pre-deployment phase, and the other is the communication overhead for forwarding the reports. The energy expenditure of EGEFS also takes into account two sources: One is the communication overhead for verifying the authenticity of locations in the reports, and the other is the communication overhead for forwarding the reports.

To be fair, we assume that all of SEF, DSF, GFFS, and EGEFS use the technique of Bloom filter. Let the length of a normal report without any extra field, node ID, location, Bloom filter, and key index be L_r, L_S, L_L, L_F, and L_k respectively. The length of the counter value C in EGEFS is denoted as L_C, and the length of the counter of recording the transmitted hops in DSF is denoted as L_h. Then, the length of a report in SEF, DSF, GFFS, and EGEFS is $Len\_SEF=L_r+T\times L_k+L_F$, $Len\_DSF=L_r+L_h+2\times T\times L_k+2\times L_F$, $Len\_GFFS=L_r+T\times(L_k+L_S+L_L)+L_F$, and $Len\_EGEFS=L_r+L_C+T\times(L_k+L_S+L_L)+L_F$ respectively. For example, when L_r=24 bytes [19, 23], T=5, L_k=10 bits, L_F=64 bits, L_S=10 bits, L_L=16 bits, L_h=10 bits, and L_C=8 bits, we can work out that *Len_SEF* is about 39 bytes, *Len_DSF* is about 54 bytes, *Len_GFFS* is about 55 bytes, and *Len_EGEFS* is about 56 bytes.

Let e_t and e_r denote the energy consumed in transmitting and receiving one byte in wireless communication respectively. When a node transmits a message, all its neighbors can hear the message and expend energy for receiving it. We assume that the number of hops that a false report travels in SEF, DSF, GFFS, and EGEFS is H_1, H_2, H_3, and H_4 respectively. The number of neighbors of each forwarding node in SEF, DSF, GFFS, and EGEFS is C_i_SEF ($1\le i\le H_1$), C_i_DSF ($1\le i\le H_2$), C_i_GFFS ($1\le i\le H_3$), and C_i_EGEFS ($1\le i\le H_4$) respectively. Then, the energy expenditure for filtering out a false report in SEF is:

$$EC\_SEF=\sum_{i=1}^{H_1} Len\_SEF\times(e_t+C_i\_SEF\times e_r) \tag{2}$$

The energy expenditure for filtering out a false report in DSF is:

$$EC\_DSF=\sum_{i=1}^{H_2} Len\_DSF\times(e_t+C_i\_DSF\times e_r) \tag{3}$$

The energy expenditure for filtering out a false report in GFFS is:

$$EC\_GFFS=\sum_{i=1}^{H_3} Len\_GFFS\times(e_t+C_i\_GFFS\times e_r) \tag{4}$$

In addition, the average energy consumption of each node for distributing the location information in GFFS is $E_{dis\_ave} = (\sum_{i=1}^{N_a} EB_i) / N_a$, where EB_i is the energy consumption of v_i for distributing its location information. The energy expenditure for filtering out a false report in EGEFS is:

$$EC\_EGEFS = \sum_{i=1}^{H_4} Len\_EGEFS \times (e_t + C_i\_EGEFS \times e_r) + EV \quad (5)$$

where EV is the communication overhead for verifying the authenticity of locations in the report, and this will be a low value because the first forwarding node which verifies the authenticity of locations is not far away from these locations.

Although *Len_EGEFS* is larger than *Len_SEF*, *Len_DSF*, and *Len_GFFS*, EGEFS still has superior performance on the aspect of energy expenditure, because EGEFS has stronger filtering ability than SEF, DSF, and GFFS, which results in $H_4 \leq H_3$, $H_4 \leq H_2$, and $H_4 \leq H_1$. Our simulation results verified this.

5.3 Storage Overhead

Here, we only consider the storage overhead caused by implementing the en-route filtering policy. In SEF, each node needs to store k keys; while in GFFS, each node needs to store the information of extra c locations (S_i, L_i, U_i), where S_i denotes the node ID, L_i denotes the location of node S_i, and U_i denotes the key partition index stored by node S_i. In DSF, each node stores k R-type keys and one A-type key. In EGEFS, each node needs to store k keys, the locations of its neighbors, and the report identifiers $\{ID_1, C\}$ of the reports which it forwards.

Denote the length of a key, the node ID, the location, U_i and C with b, L_S, L_L, L_U, and L_C respectively. Suppose there are N_a sensor nodes in the network, each of which has average C_n neighbors, and there are m normal event reports injected into the network, each of which travels average H hops to reach the sink. Then, the average storage overhead of each node in SEF, DSF, GFFS, and EGEFS is $St\_SEF=k\times b$, $St\_DSF=k\times b+b$, $St\_GFFS=k\times b+c\times(L_S+L_L+L_U)$, and $St\_EGEFS= k\times b+C_n\times L_L+ m\times(L_S+L_C)\times H/N_a$ respectively.

Obviously, SEF has the smallest average storage overhead, and whether GFFS or EGEFS has the highest average storage overhead depends on the value of parameters. For example, when k=50, b=64 bits, c=40, L_S=10 bits, L_L=16 bits, L_U=4, C_n=10, L_C=8 bits, N_a=400, H=6, and m=100, we have *St_SEF*=400 bytes, *St_DSF*=408 bytes, *St_GFFS*=550 bytes, and *St_EGEFS*=424 bytes. We can see that the average storage overhead of GFFS is highest. If we change c to 10, then the average storage overhead of GFFS is about 404 bytes, which is smaller than that of EGEFS.

6. Performance Evaluation

In this section, we further confirm the analysis in Section 5 through simulation experiments. In order to compare the performance of SEF, GFFS, DSF, and EGEFS, we conducted extensive simulations of these algorithms based on the WSN simulator in [28]. Then, we evaluated the following metrics:

1) *En-route filtering probability*, measured as the percentage of dropped false reports by the forwarding nodes. The higher the en-route filtering probability, the stronger the security performance.

2) *Number of traveled hops*, that is, the number of hops that a false report travels since it is

injected into the network until it is dropped. The smaller the number of traveled hops, the higher the filtering efficiency of the algorithm, and the lower the energy expenditure and network bandwidth.

3) *Filtering energy expenditure*, measured as the energy expenditure for filtering out a false report since it is injected into the network until it is filtered out. The lower the filtering energy expenditure is, the more beneficial it is to extend the network life.

6.1 Simulation Setting

The sensor nodes were randomly deployed in a square area of 200×200 m^2. We simulated three different scenarios according to the relationship between R_c and R_s. In Scenario 1, we set N_a=400, R_c=20, and R_s=20; in Scenario 2, we set N_a=450, R_c=20, and R_s=40; in Scenario 3, we set N_a=500, R_c=25, and R_s=12.5. In all the three scenarios, one sink and one malicious compromised node sat in opposite ends of the field. **Fig. 2** illustrates a network topology for Scenario 2, in which the square filled with blue color denotes the sink, and the circle filled with red color denotes the malicious compromised node which generates a false report and transmits it to the sink along the routing path highlighted in blue.

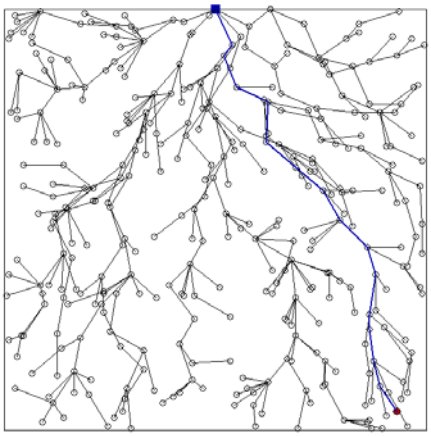


Fig. 2. An example of network topology in Scenario 2

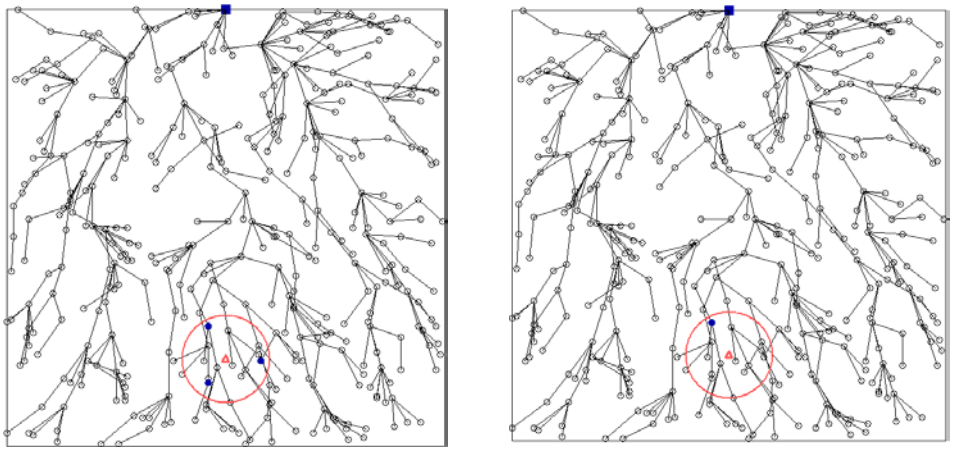
In the simulations, we used a global key pool of 1000 keys, and the pool was divided into 10 partitions, with 100 keys in each partition. Each node stored 50 keys. The number of endorsing nodes T was set to 5, and the maximum number of iterations D_{max} in DSAPS [28] was set to 50. The power consumption of transmitting and receiving 1 byte was 16.25 uJ and 12.5 uJ respectively [19]. The length of a report in SEF, GFFS, DSF, and EGEFS was 39 bytes, 55 bytes, 54 bytes, and 56 bytes respectively. The size of location message in GFFS was 5 bytes, and the sizes of ASK message and REPLY message in EGEFS were 16 bytes and 4 bytes respectively. We used RC5-32/10/8 to generate the MACs.

The performance of SEF, GFFS, and DSF is affected by the number of hops from the malicious compromised node to the sink (denoted as MNHop). The farther the malicious compromised node is from the sink, the more the forwarding nodes that the false reports pass through, and the higher the en-route filtering probability. We simulated two cases in which MNHop was 5 and 10 respectively. Furthermore, the performance of GFFS is affected by c--the number of location packets distributed by each node. The larger the c, the higher the en-route filtering probability of GFFS, but the higher the energy consumption for distributing location information. We simulated two cases in which c was 20 and 40 respectively.

6.2 Simulation Results

1) Simulation of the CoS Election Methods

As mentioned in Section 4.2, the method of CoS election in SEF/GFFS can ensure that only one CoS node will be elected when $R_c \geq 2R_s$, whereas more than one CoS node may be elected when $R_c < 2R_s$. The simulation results confirm our analysis. By using the CoS election method in SEF/GFFS, only one CoS node was elected in Scenario 3, whereas multiple CoS nodes were elected in Scenario 1 and 2. By using our CoS election method, only one CoS node was elected in all the three scenarios. **Fig. 3(a)** and **3(b)** show the results on CoS election in Scenario 1 by using the method in SEF/GFFS and our method respectively, in which the red rectangle denotes the event location, the red circle denotes the sensing range of the event, and the circles filled with blue denote the elected CoS nodes. We can see that there are three CoS nodes in **Fig. 3(a)**, and there is only one CoS node in **Fig. 3(b)**.



(a) CoS election in SEF/GFFS (b) Our CoS election

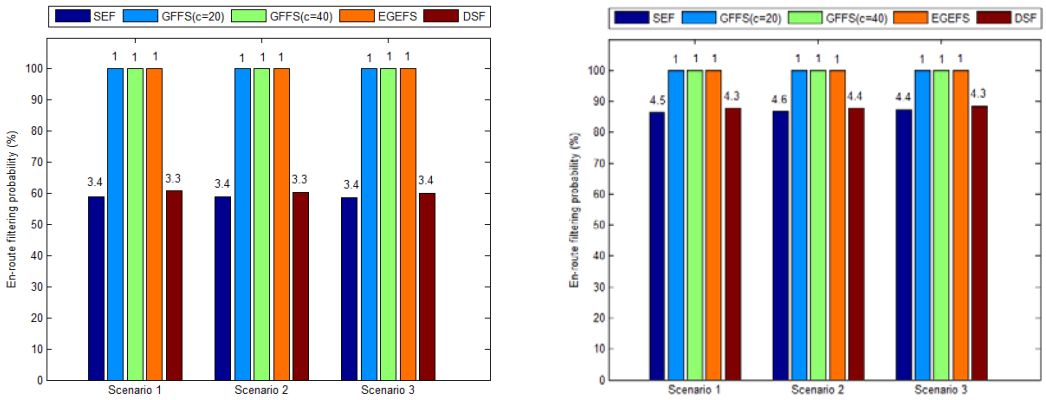
Fig. 3. Simulation results of CoS election in Scenario 1

2) Simulation of the En-route Filtering Schemes

We simulated 10 different network topologies for every scenario. In each network, the attacker controls one malicious compromised node to inject 100 bogus reports into the network. The simulation results were averaged over the 10 simulated topologies.

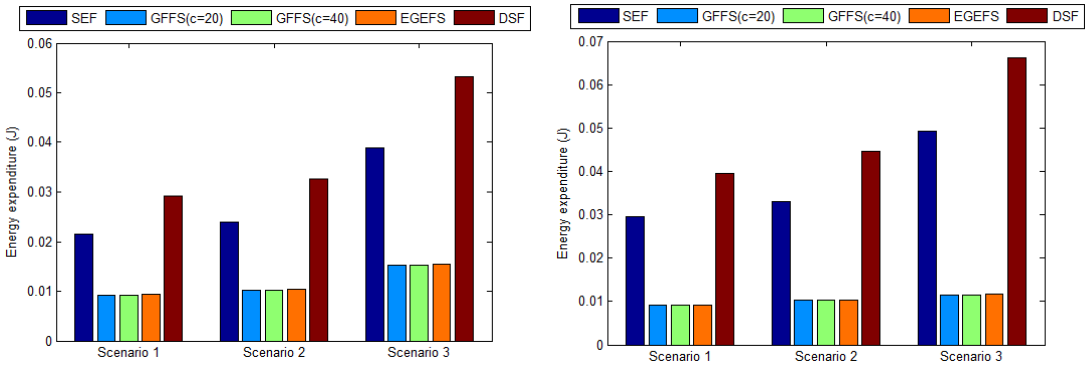
(1) Algorithm performance in Attack Model 1

Fig. 4(a) and **4(b)** show the results of the en-route filtering probability (i.e., the height of the bars) and the number of traveled hops (i.e., the value above the bars) in Attack Model 1 when MNHop=5 and MNHop=10 respectively. We observe that, the en-route filtering probabilities of GFFS and EGEFS are both 100%, which are higher than those of SEF and DSF. Furthermore, both GFFS and EGEFS can filter out false reports within one hop. However, the false reports can travel multiple hops before being detected in SEF and DSF. Therefore, both GFFS and EGEFS have stronger en-route filtering ability than SEF and DSF in Attack Model 1. For SEF and DSF, the en-route filtering probability when MNHop=10 is higher than that when MNHop=5, because the false reports can travel more hops when MNHop=10. These results are consistent with the previous analysis.



(a) MNHop=5 (b) MNHop=10

Fig. 4. En-route filtering probability and number of traveled hops in Attack Model 1



(a) MNHop=5 (b) MNHop=10

Fig. 5. Filtering energy expenditure in Attack Model 1

The results of the filtering energy expenditure in Attack Model 1 when MNHop=5 and MNHop=10 are shown in **Fig. 5(a)** and **5(b)** respectively. We can see that GFFS and EGEFS outperform SEF and DSF in filtering energy expenditures, because the false reports travel more hops in SEF and DSF. The filtering energy expenditure of DSF is higher than that of SEF although the en-route filtering probability of DSF is higher than that of SEF. This is because the length of a report in DSF is larger than that in SEF. The filtering energy expenditure of EGEFS is slightly (about 1.8%) higher than that of GFFS, because the length of a report in EGEFS is 1 byte larger than that in GFFS. However, GFFS consumes extra energy for distributing the location information. When c=20, the experimental results of the average energy consumption of each node for distributing the location information in Scenario 1, 2, and 3 are 0.138 J, 0.163 J, and 0.309 J respectively; when c=40, the corresponding experimental results are 0.289 J, 0.291 J, and 0.386 J. Therefore, overall, EGEFS is still more energy efficient than GFFS. For each algorithm, the filtering energy expenditure in Scenario 3 is the highest, and that in Scenario 1 is the lowest. This is because the average number of neighbors of a node in the three scenarios is about 11.6, 13.1, and 21.7 respectively. Then, in Scenario 3, if a node sends a message, on average, 21.7 nodes will expend energy to receive the message, which will result in high energy consumption. For SEF and DSF, the filtering energy expenditure when MNHop=10 is higher than that when MNHop=5, because the false reports travel more hops when MNHop=10.

(2) Algorithm performance in Attack Model 2

Fig. 6 shows the results of the en-route filtering probability (i.e., the height of the bars) and the number of traveled hops (i.e., the value above the bars) in Attack Model 2 when MNHop=5 or 10. The en-route filtering probabilities of SEF and DSF are both 0%, which means SEF and DSF cannot resist the attacks in Attack Model 2, and each false report reaches the sink, so the number of traveled hops of each false report is equal to the value of MNHop. The en-route filtering probabilities of GFFS and EGEFS are both 100%, and the false reports are filtered out within one hop. Obviously, both GFFS and EGEFS have strong defensive ability against Attack Model 2.

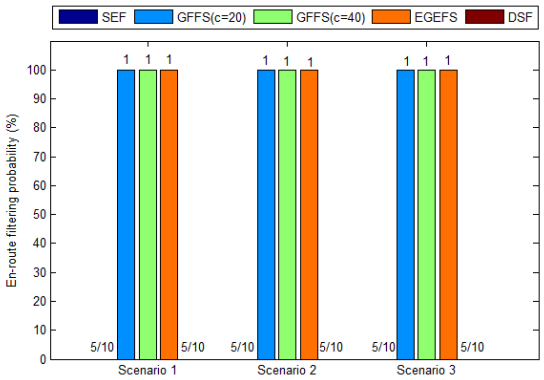
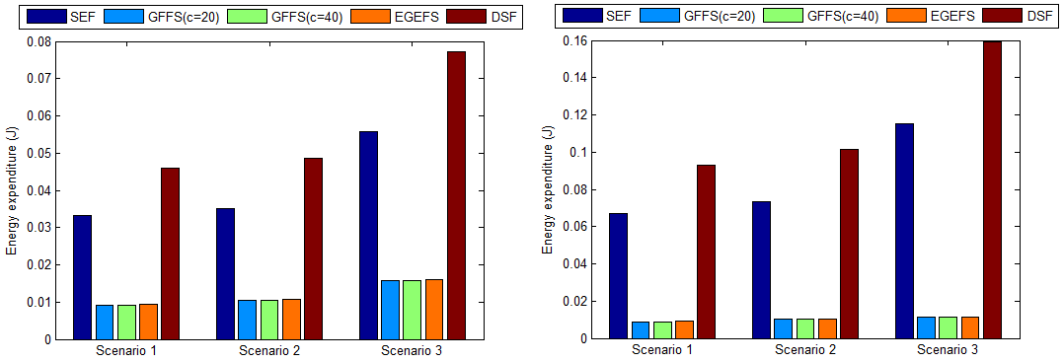


Fig. 6. En-route filtering probability and number of traveled hops in Attack Model 2 (MNHop=5 or 10)



(a) MNHop=5 (b) MNHop=10

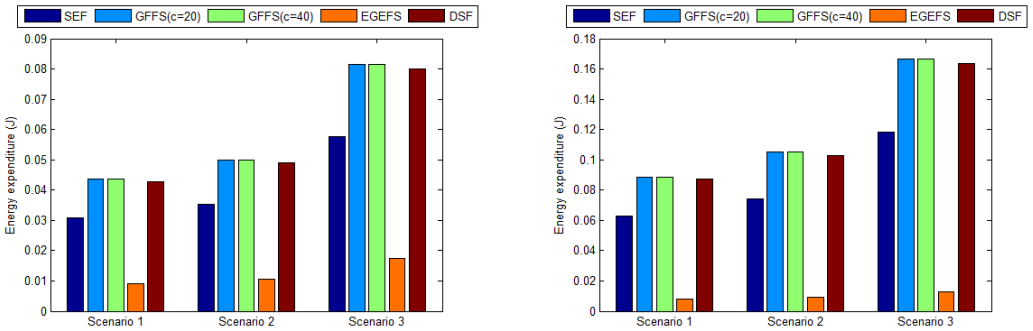
Fig. 7. Filtering energy expenditure in Attack Model 2

The results of the filtering energy expenditure in Attack Model 2 when MNHop=5 and MNHop=10 are shown in **Fig. 7(a)** and **7(b)** respectively. We can see that the filtering energy expenditure of DSF is the highest. Although the filtering energy expenditure of EGEFS is slightly higher than that of GFFS, GFFS has additional energy expenditure for distributing the location information.

(3) Algorithm performance in Attack Model 3

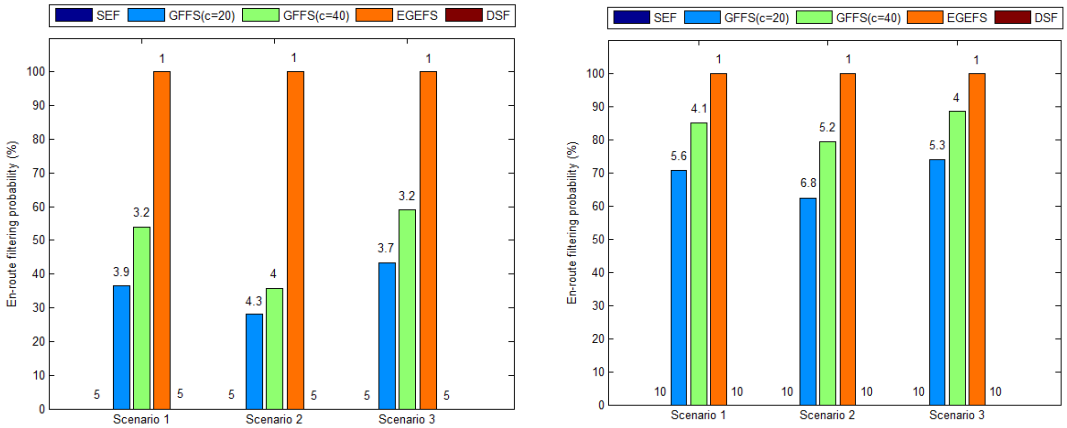
The en-route filtering probabilities of SEF, GFFS, and DSF in Attack Model 3 are all 0%, regardless of the values of *c* and MNHop, whereas the en-route filtering probability of EGEFS is 100%, which means that SEF, GFFS, and DSF cannot resist the replay attacks, but EGEFS can. In SEF, GFFS, and DSF, each duplicated report reaches the sink, so the number of traveled hops is equal to the value of MNHop. However, each duplicated report is filtered out within one hop in EGEFS.

The results of the filtering energy expenditure in Attack Model 3 when MNHop=5 and MNHop=10 are shown in **Fig. 8(a)** and **8(b)** respectively. We can see that the filtering energy expenditure of EGEFS is the lowest, because each duplicated report is filtered out at its first forwarding node. The filtering energy expenditure of GFFS is higher than that of SEF and DSF, because the length of a data report in GFFS is larger than that in SEF and DSF.



(a) MNHop=5 (b) MNHop=10

Fig. 8. Filtering energy expenditure in Attack Model 3



(a) MNHop=5 (b) MNHop=10

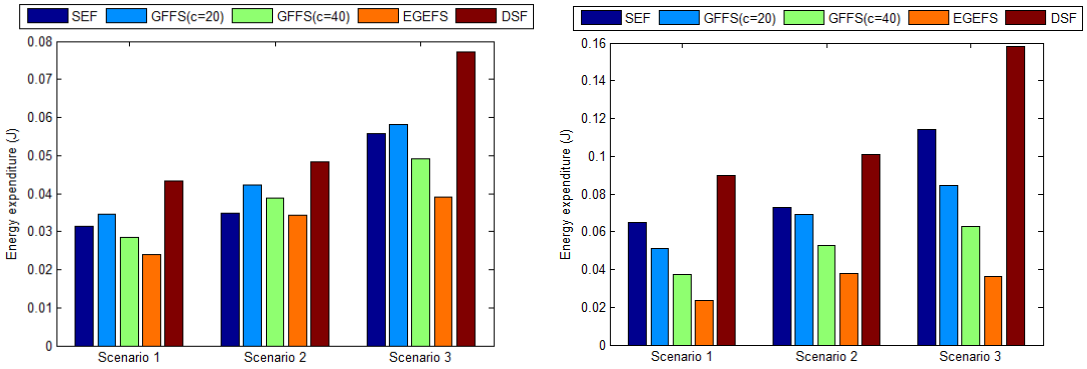
Fig. 9. En-route filtering probability and number of traveled hops in Attack Model 4

(4) Algorithm performance in Attack Model 4

Fig. 9(a) and **9(b)** show the results of the en-route filtering probability (i.e., the height of the bars) and the number of traveled hops (i.e., the value above the bars) in Attack Model 4 when MNHop=5 and MNHop=10 respectively. We observe that, the en-route filtering probabilities of SEF and DSF are both 0%, which means SEF and DSF cannot resist the attacks in Attack Model 4, so the number of traveled hops in either SEF or DSF is equal to the value of MNHop. The en-route filtering probability of GFFS is between 0% and 100%, which is affected by c, and it means GFFS has certain defensive ability against Attack Model 4. Furthermore, the en-route filtering probability when c=40 is higher than that when c=20, which confirms that the larger the c, the higher the en-route filtering probability of GFFS. The en-route filtering probability of EGEFS is 100%, and the false reports are filtered out within one hop. Therefore, EGEFS has the strongest defensive ability against Attack Model 4.

The results of the filtering energy expenditure in Attack Model 4 when MNHop=5 and MNHop=10 are shown in **Fig. 10(a)** and **10(b)** respectively. It can be seen that the filtering

energy expenditure of GFFS is lower than that of SEF in most cases, but there are exceptions. For example, as shown in **Fig. 10(a)**, the filtering energy expenditure of GFFS (c=20) in Scenario 2 is higher than that of SEF. This is because the length of a data report in GFFS is larger than that in SEF, and the number of traveled hops in GFFS (c=20) is not low enough. For GFFS, the filtering energy expenditure when c=40 is lower than that when c=20, because when c=40, the en-route filtering probability is higher, and the number of traveled hops is smaller. However, the energy expenditure for distributing the location information is higher when c=40 than when c=20. The filtering energy expenditure of EGEFS is the lowest, because the false reports are filtered out within one hop in EGEFS.



(a) MNHop=5 (b) MNHop=10

Fig. 10. Filtering energy expenditure in Attack Model 4

7. Conclusion

False data injection attack is a serious threat to wireless sensor networks. Considering that the attack models considered by the existing en-route filtering strategies are relatively simple, we present a new complicated attack model, called “collusion attack with forged locations”, and propose an efficient geographical information-based en-route filtering scheme (EGEFS), which can resist various types of false data injection attacks. In addition, we propose an effective method for electing the CoS node, which can ensure that only one detecting node will be elected as the CoS node for an event. The simulation results demonstrate that EGEFS outperforms the existing en-route filtering schemes in terms of en-route filtering probability, filtering efficiency, and energy expenditure.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61572263, 61502251, 61872197, 61402225), the Key Project of Natural Science Research of Jiangsu University (14KJB520031), the National Natural Science Foundation of Jiangsu Province (BK20151511, BK20140832), the Postdoctoral Science Foundation of China (2016M601859, 2013M540447), the Postgraduate Research & Practice Innovation Program of Jiangsu Province (KYZZ16_0259) and the Natural Research Foundation of Nanjing University of Posts and Telecommunications (NY217119).

References

[1] S. Majumder, T. Mondal and M. J. Deen, "Wearable sensors for remote health monitoring," *Sensors*, vol. 17, pp. 1-45, January, 2017. Article (CrossRef Link)

[2] R. D. Gomes, D. V. Queiroz, A. C. L. Filho, I. E. Fonseca and M. S. Alencarand, "Real-time link quality estimation for industrial wireless sensor networks using dedicated nodes," *Ad Hoc Networks*, vol. 59, pp. 116-133, May, 2017. Article (CrossRef Link)

[3] V. Bapat, P. Kale, V. Shinde, N. Deshpande and A. Shaligram, "WSN application for crop protection to divert animal intrusions in the agricultural land," *Computers & Electronics in Agriculture*, vol. 133, pp. 88-96, February, 2017. Article (CrossRef Link)

[4] J. Lee, Z. Zhong, B. Du, S. Gutesa and K. Kim, "Low-cost and energy-saving wireless sensor network for real-time urban mobility monitoring system," *Journal of Sensors*, vol. 1870, pp. 1-8, September, 2015. Article (CrossRef Link)

[5] S. Kumari and H. Om, "Authentication protocol for wireless sensor networks applications like safety monitoring in coal mines," *Computer Networks*, vol. 104, pp. 137-154, May, 2016. Article (CrossRef Link)

[6] H. W. Ferng and N. M. Khoa, "On security of wireless sensor networks: a data authentication protocol using digital signature," *Wireless Networks*, vol. 23, pp. 1113-1131, January, 2017. Article (CrossRef Link)

[7] R. Amin and G. P. Biswas, "A secure light weight scheme for user authentication and key agreement in multi-gateway based wireless sensor networks," *Ad Hoc Networks*, vol. 36, pp. 58-80, May, 2016. Article (CrossRef Link)

[8] O. Pereira, F. X. Standaert and S. V. Venkatesh, "Leakage-resilient authentication and encryption from symmetric cryptographic primitives," in *Proc. of 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 96-108, October, 2015. Article (CrossRef Link)

[9] O. R. M. Boudia, S. M. Senouci and M. Feham, "A novel secure aggregation scheme for wireless sensor networks using stateful public key cryptography," *Ad Hoc Networks*, vol. 32, pp. 98-113, January, 2015. Article (CrossRef Link)

[10] R. H. Li, J. X. Yu, X. Huang and H. Cheng, "Robust reputation-based ranking on bipartite rating networks," *SDM*, vol. 12, pp. 612-623, April, 2012. Article (CrossRef Link)

[11] M. Rezvani, A. Ignjatovic, E. Bertino, and S. Jha, "Secure data aggregation technique for wireless sensor networks in the presence of collusion attacks," *Dependable & Secure Computing IEEE Transactions on*, vol. 12, no. 1, pp. 98-110, January, 2015. Article (CrossRef Link)

[12] S. Roy, M. Conti, S. Setia and S. Jajodia, "Secure data aggregation in wireless sensor networks: filtering out the attacker's impact," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 4, pp. 681-694, April, 2014. Article (CrossRef Link)

[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, September, 2002. Article (CrossRef Link)

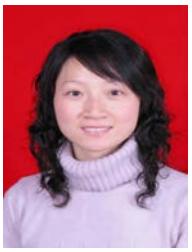
[14] H. Yang and S. Lu, "Commutative cipher based en-route filtering in wireless sensor networks," in *Proc. of IEEE 60th Vehicular Technology Conference*, pp. 1223-1227, September, 2004. Article (CrossRef Link)

[15] Y. Zhang, W. Liu, W. Lou and Y. Fang, "Location-based compromise-tolerant security mechanisms for wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 247-260, February, 2006. Article (CrossRef Link)

[16] H. Wang and Q. Li, "Achieving robust message authentication in sensor networks: a public-key based approach," *Wireless Networks*, vol. 16, no. 4, pp. 999-1009, May, 2010. Article (CrossRef Link)

[17] H. Yu and J. He, "Authentication and en-route data filtering for wireless sensor networks in the internet of things scenario," *International Journal of Grid and Distributed Computing*, vol. 6, no. 1, pp. 1-12, February, 2013. Article (CrossRef Link)

[18] M. K. Shahzad and T. H. Cho, "An energy-aware routing and filtering node (ERF) selection in CCEF to extend network lifetime in WSN," *IETE Journal of Research*, vol. 63, pp. 368-380, February, 2017. Article (CrossRef Link)
[19] F. Ye, H. Luo, S. Lu and L. Zhang, "Statistical en-route filtering of injected false data in sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 839-850, April, 2005. Article (CrossRef Link)
[20] Z. Yu and Y. Guan, "A dynamic en-route scheme for filtering false data injection in wireless sensor networks," in *Proc. of 3rd International Conference on Embedded Networked Sensor Systems*, pp. 1-12, November, 2005. Article (CrossRef Link)
[21] J. Li, L. Yu, H. Gao and S. Xiong, "Grouping-enhanced resilient probabilistic en-route filtering of injected false data in wsns," *IEEE Transactions on Parallel & Distributed Systems*, vol. 23, no. 5, pp. 881-889, May, 2012. Article (CrossRef Link)
[22] R. Lu, X. Lin, H. Zhu, X. Liang and X. Shen, "BECAN: a bandwidth-efficient cooperative authentication scheme for filtering injected false data in wireless sensor networks," *IEEE Transactions on Parallel & Distributed Systems*, vol. 23, no. 1, pp. 32-43, January, 2012. Article (CrossRef Link)
[23] J. Wang, Z. Liu, S. Zhang and X. Zhang, "Defending collaborative false data injection attacks in wireless sensor networks," *Information Sciences*, vol. 254, no. 1, pp. 39-53, January, 2014. Article (CrossRef Link)
[24] Q. Sun and M. Wu, "A double key-sharing based false data filtering scheme in wireless sensor networks," *Journal of Computers*, vol. 8, no. 2, pp. 509-516, February, 2013. Article (CrossRef Link)
[25] H. Xu, H. Sun, Y. Cheng and H. Liu, "Wireless sensor networks localization based on graph embedding with polynomial mapping," *Computer Networks*, vol. 106, pp. 151-160, June, 2016. Article (CrossRef Link)
[26] X. Fang, Z. Jiang, L. Nan and L. Chen, "Noise-aware localization algorithms for wireless sensor networks based on multidimensional scaling and adaptive Kalman filtering," *Computer Communications*, vol. 101, pp. 57-68, March, 2017. Article (CrossRef Link)
[27] A. Stanoev, S. Filiposka, V. In and L. Kocarev, "Cooperative method for wireless sensor network localization," *Ad Hoc Networks*, vol. 40, pp. 61-72, January, 2016. Article (CrossRef Link)
[28] C. J. Yi, G. Yang, H. Dai, L. Liu and N. Li, "Switching algorithm with prediction strategy for maximizing lifetime in wireless sensor network," *International Journal of Distributed Sensor Networks*, pp. 1-12, November, 2015. Article (CrossRef Link)
[29] R. L. Rivest, "The RC5 encryption algorithm," in *Proc. of Workshop on Fast Software Encryption*, pp. 86-96, December, 1994. Article (CrossRef Link)
[30] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Proc. of 6th Annual International Conference on Mobile Computing and Networking*, pp. 243-254, August, 2000. Article (CrossRef Link)



Chuanjun Yi, born in 1981, Ph.D. student of Nanjing University of Posts and Telecommunications, lecturer of Zijin College, Nanjing University of Science and Technology. Her research interests include graphics and image processing, information security, and wireless sensor network. E-mail: zjyicj@126.com.

Geng Yang, born in 1961, professor and PhD supervisor at the College of Computer Science and Technology, Nanjing University of Posts and Telecommunications. His current research interests include computer network, parallel and distributed computing, wireless sensor network, and information security. E-mail: yangg@njupt.edu.cn.

Hua Dai, born in 1982, PhD and associate professor at the College of Computer Science and Technology, Nanjing University of Posts and Telecommunications. His current research interests include database security, distributed data management and security. E-mail: daihua@njupt.edu.cn.

Liang Liu, born in 1985, PhD and lecturer at the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics. His research interests include issues related to data security, cloud data management, and wireless sensor network. E-mail: liangliu@nuaa.edu.cn.

Yunhua Chen, born in 1980, lecturer of Nanjing Institute of Mechatronic Technology. Her research interests include graphics and image processing, information security, and wireless sensor network. E-mail: chenyh1011@163.com.