Research Article

En-Route Message Authentication Scheme for Filtering False Data in WSNs

Chuanjun Yi

School of Computer Engineering, Jinling Institute of Technology, Nanjing 211169, China

Correspondence should be addressed to Chuanjun Yi; zjyicj@126.com

Received 26 April 2021; Revised 4 August 2021; Accepted 25 August 2021; Published 11 September 2021

Academic Editor: Shah Nazir

Copyright © 2021 Chuanjun Yi. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In wireless sensor networks, the adversary can easily control the compromised nodes to inject false data reports. En-route filtering is an effective mechanism to resist such attacks, where the forwarding nodes of the reports can identify and drop the false reports. However, the existing en-route filtering strategies are vulnerable to report disruption attacks and selective forwarding attacks, and the probabilities and efficiencies of en-route filtering false reports are low. To address these problems, a precheck mechanism performed by the CoS (Center-of-Stimulus) node is presented to resist report disruption attacks, a report forwarding strategy with balancing the residual energy of the nodes is designed to resist selective forwarding attacks, and an en-route message authentication scheme (EMAS) based on monitoring and reporting mechanism is proposed to resist false data injection attacks. The theoretical analysis and simulation results show that in most cases, EMAS provides a higher security level and higher en-route filtering probability and efficiency and is very efficient in energy saving.

1. Introduction

Wireless sensor networks (WSNs) have been applied in various applications, including military application, industrial monitoring [1, 2], agricultural monitoring [3], and health care [4]. In these applications, a large number of sensor nodes with limited resources are deployed to detect events of interest and deliver data reports to the sink via multihop wireless communication. WSNs are often deployed in unattended or even hostile environments; therefore, the sensor nodes in WSNs are vulnerable. The adversary can compromise sensor nodes by means of node replication attacks and code injection attacks. Then, the adversary can control these compromised nodes to launch various outside attacks, including physical destruction of sensor nodes, security attacks on the routing and data link protocols, and resource consumption attacks. Besides, the adversary can use these compromised nodes to launch various insider attacks, including false data injection attacks, selective forwarding attacks, and report disruption attacks. These insider attacks can disrupt network function, induce network congestion, and waste network resources (e.g., energy, bandwidth, and storage space).

In order to resist false data injection attacks and filter out false data reports as early as possible, scholars have proposed a number of en-route filtering strategies, in which the forwarding nodes verify the data reports and discard the false ones. According to the adopted data encryption technology, the existing en-route filtering strategies can be classified into symmetric cryptography-based strategies [5–14] and asymmetric cryptography-based strategies [15–22]. Among them, the symmetric cryptography-based strategies attract more attention because of their advantages in communication overhead, computation overhead, and storage overhead.

However, the existing symmetric cryptography-based schemes have some drawbacks. For example, SEF [5], PCREF [6], CFPS [7], GFPS [8], EGEFS [9], and NHFS [14] are vulnerable to report disruption attacks and selective forwarding attacks. Besides, most of the existing en-route filtering schemes adopt simple verification methods, such as MAC (message authentication code) verification, resulting in low probability and efficiency of en-route filtering false reports. To overcome these drawbacks, some effective techniques are proposed to resist various attacks, including false data injection attacks, report disruption attacks, and

selective forwarding attacks, so as to improve the system security and the en-route filtering probability and efficiency. The major contributions of this paper are outlined as follows:

- (1) In order to resist report disruption attacks, a pre-check mechanism performed by the CoS (Center-of-Stimulus) node is proposed. Before generating the event report, the CoS node checks the endorsements provided by the detection nodes and discards the illegitimate ones, thereby ensuring the correctness of the endorsements in the event report.
- (2) In order to defend against selective forwarding attacks, a report forwarding strategy with balancing the residual energy of the sensor nodes is proposed. When selecting a forwarding node, the sender prefers the upstream neighbor (the neighbor closer to the sink) with more residual energy, so as to balance the residual energy of nodes in the network and prolong the network life. If the report is not forwarded, the sender will try other upstream neighbors until the report is forwarded.
- (3) An en-route message authentication scheme (EMAS) based on monitoring and reporting mechanism for filtering false reports is proposed. EMAS improves the en-route filtering probability and efficiency by verifying more information carried in the event report (i.e., MACs, the IDs and locations of endorsing nodes, and the prev) and by using the monitoring and reporting mechanism. The theoretical analysis and simulation results demonstrate that, in most cases, EMAS outperforms SEF, GFFS, and EGEFS in terms of security, en-route filtering probability, en-route filtering efficiency, and energy expenditure.

The rest of this paper is organized as follows. In Section 2, the related works in the area of en-route filtering false reports are introduced. Section 3 presents the system models and threat models. The proposed scheme is detailed in Section 4, and its performance is analyzed and evaluated in Sections 5 and 6, respectively. Finally, Section 7 concludes this paper.

2. Related Works

The existing en-route filtering strategies include symmetric cryptography-based strategies and asymmetric cryptography-based strategies. Due to the limited space, only the existing symmetric cryptography-based strategies that are more relevant to this research will be discussed.

Ye et al. [5] present a statistical en-route filtering (SEF) mechanism for forwarding nodes to identify and drop false reports. In SEF, each detecting node generates a keyed MAC for the event, and T ($T > 1$) MACs are attached to the event report. The forwarding nodes verify the correctness of the MACs in the report with certain probability. When any invalid MAC is found, the report will be discarded. The simulation results show that SEF can drop up to 70% of false reports within five hops. However, a few false reports with

incorrect MACs may escape en-route filtering and reach the sink. Furthermore, SEF can only tolerate a small number of compromised nodes.

Yang et al. [6] proposed a polynomial-based, compromised, resilient en-route filtering scheme (PCREF), which adopts polynomials in place of MACs to endorse and verify reports. In PCREF, each node stores two types of polynomials: authentication polynomial and check polynomial derived from the primitive polynomial. PCREF can filter out false data effectively and achieve high resilience to the large number of compromised nodes without relying on fixed routes for data transmission. However, PCREF is prone to selective forwarding attacks and report disruption attacks, and it has T -threshold limitation.

Liu et al. [7] proposed a cluster-based false data filtering scheme (CFFS) for filtering false data efficiently. The nodes are organized into clusters and a sink-rooted tree of cluster heads is constructed. In addition, a distributed key assignment method is proposed, which assigns keys to upstream nodes based on the tree-like path to sink. Thus, false reports can be verified by several nodes during one hop. CFFS outperforms the existing schemes in terms of filtering efficiency and overhead balance. However, it is prone to T -threshold limitation, selective forwarding attacks, and report disruption attacks, and it cannot adapt to dynamic networks.

Wang et al. [8] considered a new type of false data injection attack called collaborative false data injection and proposed a geographical information-based false data filtering scheme (GFFS). In the predeployment phase, each node needs to distribute its location and key partition to the intermediate nodes. Each event report carries t ($t > 1$) MACs and locations. The forwarding nodes verify the correctness of MACs and locations and the legitimacy of the locations. GFFS can filter out the false reports injected collaboratively by compromised nodes. However, distributing the information of location and key partition incurs long latency and high-energy overhead.

Yi et al. [9] designed a new type of false data injection attack called collusion attack with forged locations and proposed an efficient geographical information-based en-route filtering scheme (EGEFS). In EGEFS, the MACs, the report identifier, and the legitimacy and authenticity of endorsing nodes' locations in the event report will be verified by the forwarding nodes. EGEFS can resist various types of false data injection attacks and can perform better in terms of en-route filtering probability, en-route filtering efficiency, and energy overhead. However, EGEFS is prone to selective forwarding attacks.

Kumar and Pais [10] proposed a multisink en-route filtering mechanism where the network is divided into smaller networks, and a separate sink is assigned to each smaller network. This helps reduce the overall energy consumption of the network. They also proposed a novel deterministic en-route filtering scheme [11] and a partial key predistribution-based scheme [12], both of which assign the secret keys to cluster heads based on combinatorial design. These two schemes provide more effective en-route filtering of false reports. However, three copies of each event report with different endorsements forwarded in the network

incurs high-energy expenditure. In addition, they propose a blockchain based deterministic en-route filtering scheme [13] which can also adapt to dynamic networks and mobile sinks.

Liu and Liu [14] proposed a neighbor information and one-way hash chain-based filtering scheme (NHFS), which binds the keys of sensor nodes to their geographical locations. Each report must carry the MACs and hash values from t detecting nodes. The forwarding nodes can filter out false reports by checking the correctness of the MACs, hash values, and the freshness of these hash values. However, NHFS is vulnerable to selective forwarding attacks and report disruption attacks.

It can be seen that the filtering technologies used in the existing en-route filtering strategies are weak, resulting in low en-route filtering probability and efficiency. In addition, most of them are vulnerable to selective forwarding attacks and report disruption attacks. In this paper, the en-route filtering scheme that can resist false data injection attacks, report disruption attacks, and selective forwarding attacks will be researched to improve the en-route filtering probability and efficiency, thereby saving the limited network resources.

3. System Models and Threat Models

In this section, system models and threat models will be introduced.

3.1. System Models. Suppose that q sensor nodes $\{v_1, \dots, v_q\}$ and one sink node are randomly deployed in the monitoring area. All the sensor nodes have the same communication range R_c and same sensing range R_s . Typically, $R_c \geq 2R_s$ [16]. After deployment, the sink and all the sensor nodes are static, and each node will obtain its location through GPS or the localization algorithms [23, 24].

Suppose that the sensor nodes in the network are dense enough so that each event can be detected simultaneously by multiple nodes. In order to make it difficult for the adversary to forge event reports, each event report needs to contain the endorsements of T ($T \geq 1$) detecting nodes with different key partitions (called *endorsing nodes*). When an event occurs, each detecting node generates a MAC and sends it to the elected CoS node. The CoS node randomly chooses T MACs from the detecting nodes with different key partitions to produce an event report and then forwards the event report to the sink through multiple hops.

The system model considered in this paper is shown in Figure 1. The small black circles denote the sensor nodes, the square filled with blue color denotes the sink, the red triangle denotes the location of the event, the red circle denotes the sensing range of the event, and the small circle filled with blue color is the elected CoS node. Each detecting node sends its endorsement of the event to the CoS node. After generating the event report, the CoS node sends it to the sink along the routing path highlighted in red.

3.2. Threat Models. Assume that the sink will not be compromised by the adversary, while other sensor nodes can be physically captured and compromised. After compromising

FIGURE 1: System model.

a node, the adversary can obtain its related information (e.g., ID, location, key, and key index) and can control it to launch various attacks. The compromised node can pretend to be a CoS node, forge false events occurring around it, and generate false event reports with legitimate forms using the information (e.g., key) of other compromised nodes. The compromised node can also pretend to be a forwarding node to inject false reports into the network. Furthermore, the adversary can control the compromised nodes to launch selective forwarding attacks. In other words, the compromised nodes may selectively discard the received reports; thus, some legitimate reports cannot reach the sink, which will severely damage data availability and disrupt the event report service. The adversary can also control the compromised nodes to launch report disruption attacks by providing illegitimate endorsements for event reports, causing the reports of real events to be dropped by some forwarding nodes or the sink.

4. The Proposed Scheme

In this section, the design overview of the proposed scheme will be described first. Then, the process of predeployment and initialization, report generation and forwarding, en-route filtering, and sink verification will be introduced in turn.

4.1. Overview of Algorithm Design. The design overview of the proposed scheme will be described here, including the design of en-route filtering strategy, the design of method for resisting report disruption attacks, and the design of method for resisting selective forwarding attacks.

4.1.1. Design of En-Route Filtering Strategy. SEF [5] adopts MAC verification to filter out false reports. However, the malicious compromised node can utilize the keys of other compromised nodes to forge MACs in order to pass the MAC verification of SEF, which results in low en-route filtering probability. In order to effectively filter out the false reports collaboratively forged by multiple compromised nodes, GFFS [8] provides a method of verifying the legitimacy of the endorsing nodes' locations. However, if the adversary forges the locations of endorsing nodes within the sensing range of the forged event, it will smoothly pass the legitimacy verification of the endorsing nodes' locations by

GFFS. In order to defend against such collusion attack by forging locations, a method of verifying the IDs of endorsing nodes is designed in this paper, which will be executed by the 1-hop forwarding node of the CoS node. Suppose that the malicious node v_i pretends to be a CoS node, it forges an event report R by using the information of other compromised nodes and then sends R to its 1-hop forwarding node v_j . Because the endorsing nodes in R should detect the same event as v_i , the distance between each endorsing node and v_i should be less than or equal to $2R_s$. Usually, $R_c \geq 2R_s$ [16], then the distance between each endorsing node and v_i will be less than or equal to R_c , which means that each endorsing node should be a neighbor of v_i . Therefore, v_j can check the neighbor ID list of v_i stored locally, and if any endorsing node in R is not in the neighbor ID list of v_i , v_j will drop R .

In order to pass the above verification of the endorsing nodes' IDs by v_j , v_i might fraudulently use the IDs of its neighbors. To combat such attack, a monitoring and reporting mechanism is proposed. When the endorsing node v_a sends its endorsement to v_i , v_a and the common neighbors of v_a and v_i (called the monitoring nodes of v_a and v_i) will store the transmission record of this endorsement. Assume that v_i fraudulently uses the ID of its neighbor v_t as the endorsing node ID to forge the report R and sends R to v_j . When v_t or a common neighbor of v_i and v_t hears R , it will check the local endorsement transmission list, and if no record of v_t sending its endorsement to v_i is found, it will send an ALERT message to v_j to report v_i . When the number of valid ALERT messages for v_i exceeds the preset threshold, v_j will drop R .

In order to pass the verification of the endorsing nodes' IDs and avoid being reported by the monitoring nodes, v_i might pretend to be a forwarding node and inject false reports into the network. In order to defend against such attack, a *prev* field can be added to the head of the event report R to record the ID of previous hop node. When v_i pretends to be a forwarding node of R and uses the ID of a nonneighbor node as the *prev* in R , then after receiving R from v_i , v_j only needs to check the neighbor ID list of v_i stored locally, and if v_j finds that the node *prev* is not the neighbor of v_i , it will drop R .

In order to pass the above *prev* verification, v_i might use a neighbor's ID as the *prev*. The monitoring and reporting mechanism can resist such attack. When v_x sends an event report to v_y , both v_x and the common neighbors of v_x and v_y will store the transmission record of this report. If v_i uses the ID of its neighbor v_k as the *prev* to forge the report R and sends R to v_j , then after v_k or a common neighbor of v_k and v_i hears R , it will check the local report transmission list. If no record of v_k sending R to v_i is found, an ALERT message will be sent to v_j to report v_i . When the number of valid ALERT messages for v_i exceeds the preset threshold, v_j will drop R .

4.1.2. Design of the Method for Resisting Report Disruption Attacks. The compromised nodes may launch report disruption attacks by providing false endorsements for the event. If the CoS node selects false endorsements to generate

the event report, then the event report will be dropped by a forwarding node and cannot reach the sink. In order to resist such attack, a precheck mechanism for the CoS node is presented. When receiving an endorsement from a detecting node, the CoS node will verify the endorsement and discard the illegitimate one. In this way, the correctness of endorsements in the event report is guaranteed.

4.1.3. Design of the Method for Resisting Selective Forwarding Attacks. In order to resist selective forwarding attacks and balance the residual energy of nodes in the network to prolong the network life, an effective report forwarding strategy is designed in this paper. When choosing a forwarding node, the sender will prefer the upstream neighbor with more residual energy. If the chosen upstream neighbor does not forward the report, the sender will choose another upstream neighbor with more residual energy to forward the report. Unless all the upstream neighbors of the sender have been compromised, the legitimate report will be ultimately forwarded to the sink.

4.2. Predeployment and Initialization. Before deployment, key assignment and storage for each sensor node should be completed. As in SEF [5], a global key pool containing N keys with different indexes is generated and divided into n ($n > T$) nonoverlapping partitions with m keys in each partition. The user randomly picks one partition for each node and stores any k ($k < m$) keys of this partition and the associated key indexes into the node. The sink holds the global key pool and knows the secret information of each node.

After network deployment, each node calculates its level (its distance to the sink) using the method described by Yi et al. [25] and then broadcasts its ID, location, level, residual energy, key partition, and its neighbors' IDs to its neighbors. Next, each node sends its location to the sink. As a result, each node stores the ID, location, level, residual energy, and key partition of all its neighbors, as well as the neighbors' IDs of each neighbor.

4.3. Report Generation and Forwarding. When an event happens, an event report will be generated and forwarded to the sink. Next, how to generate and forward an event report will be detailed.

4.3.1. Report Generation. When an event occurs, a CoS node should be selected from the detecting nodes to generate the event report and send it to the sink. The CoS selection method in SEF [5] is adopted in this paper.

After the CoS node is selected, each detecting node v_j randomly selects a locally stored key K_i (i is the key index of K_i) and generates a MAC: $M_i = \overline{\text{MAC}}(K_i, \text{ID}_{\text{CoS}} \| C \| L_E \| E)$, where $\|$ denotes stream concatenation, L_E and E denote the location and reading of the event, and $\{\text{ID}_{\text{CoS}}, C\}$ denotes the report identifier, as in EGEFS [9]. The single-block encryption algorithm RC5 is adopted to compute the MAC. Then, v_j sends its endorsement of the event, i.e., $\{\text{ID}_{\text{CoS}}, C,$

$ID_{vj}, L_j, i, M_i\}$, to the CoS node. The CoS node, v_j , and the common neighbors of the CoS node and v_j store the transmission record of this endorsement (including $\{ID_{vj}, ID_{cos}, C\}$) into the local endorsement transmission list ETLT.

After receiving the endorsement from v_j , the CoS node will verify this endorsement and discard it if the verification fails, as shown in the following.

First of all, the CoS node verifies whether L_j and the key partition of i are consistent with the corresponding information of v_j stored locally, and if not, the endorsement will be discarded; otherwise, the CoS node checks whether the key partition of v_j is the same as that of a selected endorsing node, and if it is, the endorsement will be discarded; otherwise, the CoS node checks whether $|L_E, L_j| \leq R_s$, and if not, the endorsement will be discarded; otherwise, the CoS node selects v_j as an endorsing node. After the number of selected endorsing nodes (including the CoS node) reaches T , the CoS node will no longer process the endorsements sent by other detecting nodes.

After selecting T endorsing nodes, the CoS node attaches the T endorsements of the endorsing nodes to the event report. In order to reduce the length of event report, the technique of Bloom filter described by Ye et al. [5] is adopted. The T MACs are mapped into a d -bit string: $F = b_0b_1, \dots, b_{d-1}$, using z independent hash functions. In this way, the final event report R looks like $\{L_E, t_E, E, C, prev, ID_1, L_1, i_1, \dots, ID_T, L_T, i_T, F\}$, where t_E refers to the time when the event is detected, ID_1 is ID_{Cos} , $prev$ is the ID of the previous hop node (the CoS node sets the $prev$ as -1 , and each forwarding node sets the $prev$ as the ID of its previous hop node before forwarding R). After generating the event report, the CoS node sends it to the next hop according to the report forwarding strategy detailed later.

4.3.2. Report Forwarding. When designing the report forwarding strategy, the main design concepts include the following: (1) The sender will select an upstream neighbor as the forwarding node to ensure that the path for forwarding the report to the sink is the shortest, thereby reducing the energy consumption for forwarding the report. (2) The sender will select the upstream neighbor with higher residual energy as the forwarding node, so as to balance the residual energy of nodes in the network and extend the network life. However, the upstream neighbor with highest residual energy should not always be selected; otherwise, it will become an explicit attack target for the adversary. (3) In order to resist selective forwarding attacks, the sender will select another upstream neighbor to forward the report if the selected upstream neighbor does not forward the report. In conclusion, the proposed report forwarding strategy is detailed as follows.

When a node v_i needs to send a report to the sink, it will select kn neighbors with highest residual energy from its upstream neighbors. Then, it will randomly choose one from the kn neighbors as the forwarding node. If the number of upstream neighbors is less than kn , then v_i will randomly choose one from the upstream neighbors as the forwarding node. Assume that v_i chooses v_j as the forwarding node and

sends the report to v_j . All the upstream neighbors of v_i will temporarily store this report until it is forwarded or dropped. Then, v_i will overhear the channel for a while. If v_i finds that v_j does not forward the report, another upstream neighbor will be chosen according to the above selection strategy to forward the report. Unless all the upstream neighbors of v_i are compromised, the legitimate report will ultimately be forwarded to the sink.

4.4. En-Route Filtering. When v_i sends an event report R to the forwarding node v_j , v_j (the specified receiver of R) and the neighbor of v_i who hears R (the nonspecified receiver of R) will conduct different processing of R :

- (1) After hearing R , v_i 's neighbor v_t (the nonspecified receiver of R) will conduct the following processing operation (the pseudocode of processing operation is described in Algorithm 1).

If v_t is a common neighbor of v_i and v_j , it will add the report transmission record $\{v_i, v_j, ID_{cos}, C\}$ to the local report transmission list RTLT. If v_i is the CoS node in R and v_t is an endorsing node v_a in R or a common neighbor of v_i and an endorsing node v_a in R , v_t will search the local endorsement transmission list ETLT. If v_t does not find the record of v_a sending its endorsement of the event to v_i , it will send an ALERT message to v_j for reporting v_i , which contains $\{ID_{vt}$ (the reporter's ID), ID_{vj} (the receiver's ID), 0 (the report type), ID_{cos} , C , and ID_{vi} (the ID of the reported node)}. If v_i is not the CoS node in R and v_t is the node $prev$ or a common neighbor of v_i and the node $prev$ in R , v_t will search the local report transmission list RTLT. If v_t does not find the record of the node $prev$ sending R to v_i , it will send an ALERT message to v_j for reporting v_i with the report type set to 1. Note: In order to reduce the communication overhead, before sending the ALERT message to v_j , if v_t has heard more than s same-type (0-type or 1-type) ALERT messages for v_i , it will cancel sending the ALERT message.

After receiving the ALERT message, v_j will verify whether the reporter v_t in the ALERT message is a neighbor of the reported node v_i , and if not, the ALERT message will be discarded; otherwise, the ALERT message will be considered to be valid, and according to the report type x ($x=0$ or 1) in the ALERT message, v_j will increase the number of x -type ALERT messages for v_i (denoted as $count_ix$) by 1.

Furthermore, another problem needs to be addressed: the reporter v_t that sends the ALERT message to v_j is not necessarily a neighbor (i.e., 1-hop neighbor) of v_j , or it may be a 2-hop neighbor (i.e., a neighbor's neighbor) of v_j , then how does v_t transmit the ALERT message to v_j ? A simple ALERT message transmission method is designed in this study. If v_j is a neighbor of v_t , v_t will directly

unicast the ALERT message to v_j ; otherwise, v_t will broadcast the ALERT message, and if a node v_f receiving this broadcast ALERT message detects that v_j as its neighbor, it will unicast the ALERT message to v_j ; otherwise, it will not forward this ALERT message. In order to reduce energy consumption, if v_f finds that another node has unicasted this ALERT message to v_j , it will not unicast this ALERT message to v_j .

In conclusion, the pseudocode for processing the ALERT message is shown in Algorithm 2.

- (2) After receiving R from v_i , the specified receiver v_j will verify R according to the following steps.

Step 1: initialize the numbers of 0-type and 1-type ALERT messages for v_i (i.e., count_i0 and count_i1) to 0. Then, start a timer T_a during which v_j receives and processes the ALERT messages for v_i . When the timer T_a expires, execute Algorithm 3, and if the returned value of Algorithm 3 is 1, discard R .

Step 2: check the freshness of R according to t_E in R and discard R if it is an overdue report.

Step 3: check the format of R is complete, the key indexes belong to different key partitions, and there are no more than $z \times T$ "1"s in F ; discard R otherwise.

Step 4: check the locations of endorsing nodes in R is legitimate, i.e., $|L_E, L_i| \leq R_s$ ($1 \leq i \leq T$); discard R otherwise.

Step 5: if v_j has a key $K \in \{K_i, 1 \leq i \leq T\}$, it calculates $M = \overline{\text{MAC}}(K, \text{ID}_{\text{CoS}} \| C \| L_E \| E)$ and the z hash values of M , then checks whether the corresponding bits in F are "1"s. If not, v_j discards R .

Step 6: if v_i is the CoS node in R , v_j checks the neighbor ID list of v_i stored locally, and if any endorsing node in R is found not to be a neighbor of v_i , v_j will discard R . If v_i is not the CoS node in R , v_j verifies the prev in R . If the prev is found not in the neighbor ID list of v_i , v_j will discard R .

If the report R has passed all the above checks, v_j will set the prev in R to v_x , then send R to the next hop v_i according to the proposed report forwarding strategy, and add the report transmission record $\{v_j, v_x, \text{ID}_{\text{CoS}}, C\}$ to the local report transmission list RTL_T.

4.5. Sink Verification. In the worst case, a false report may escape en-route filtering by all the forwarding nodes and reach the sink. At this time, the sink will verify the report and discard the false one, thereby avoiding making wrong decisions.

When the sink receives an event report R from v_k , it first verifies R according to Step 2, Step 3, and Step 4 in Section 4.4. Then, it checks whether $\{\text{ID}_j, L_j, i_j\}$ ($1 \leq j \leq T$) in R are the same as the corresponding information stored locally. Next, for each key $K \in \{K_i, 1 \leq i \leq T\}$, it calculates $M = \overline{\text{MAC}}(K, \text{ID}_{\text{CoS}} \| C \| L_E \| E)$, then regenerates the Bloom filter F' and checks whether $F' = F$. After that, it checks whether the endorsing nodes in R are all neighbors of the

CoS node and the node prev in R is a neighbor of v_k . Finally, it checks whether the numbers of 0-type and 1-type ALERT messages for v_k are both less than s . If any of the above checks fails, the report R will be discarded.

5. Performance Analysis

In this section, the security features, communication overhead, and storage overhead for SEF [5], GFFS [8], EGEFS [9], and EMAS will be discussed.

5.1. Security Features. In order to evaluate the en-route filtering capabilities of SEF [5], GFFS [8], EGEFS [9], and EMAS, four types of false data injection attacks are designed as follows:

- (1) Collusion attack by forging locations.

The malicious compromised node v_i forges an event occurred around it and acts as the CoS node to forge a report for the event in a legitimate form. First, v_i chooses T compromised nodes with different key partitions as the endorsing nodes of the forged event and then forges their locations within the sensing range of the forged event. When the number of compromised nodes with different key partitions is less than T , the information (e.g., IDs, key indices, locations, and MACs) of the rest required endorsing nodes will be forged. Among them, the IDs and MACs are forged randomly, the key indices are forged to belong to different key partitions from those of other endorsing nodes, and the locations are forged to be within the sensing range of the forged event. Then, v_i sets the prev in the report as -1 . After generating the forged report, v_i sends it to its 1-hop forwarding node.

For convenience of reference, the collusion attack by forging locations is denoted as the CAFL attack.

- (2) Collusion attack by fraudulently using neighbors' information.

The malicious compromised node v_i forges an event nearby, making as many neighbors with different key partitions as possible within the sensing range of the forged event. Then, v_i acts as the CoS node to forge a report for the event in a legitimate form, using the IDs and locations of the neighbors with different key partitions within the sensing range of the forged event, as well as the keys and key indices of the compromised nodes with the same key partitions as the selected neighbors. When there are insufficient available neighbors or insufficient available compromised nodes with different key partitions, v_i will forge the rest required information of endorsing nodes by using the method in the CAFL attack. Then, v_i sets the prev in the report as -1 . After generating the forged report, v_i sends it to its 1-hop forwarding node.

```

/*When  $v_t$  hears the report  $R$  sent by  $v_i$  to  $v_j$  */
if ( $v_t$  is a common neighbor of  $v_i$  and  $v_j$ )
    Add the report transmission record  $\{v_i, v_j, ID_{cos}, C\}$  to the local list RTL $T$ ;
if ( $v_i$  is the CoS node in  $R$ ) {
    if ( $v_t$  is an endorsing node  $v_a$  in  $R$  ||  $v_t$  is a common neighbor of  $v_i$  and an endorsing node  $v_a$  in  $R$ ) {
        Search the local list ETL $T$  for the record of  $v_a$  sending its endorsement of the event to  $v_i$ .
        If not found, set  $flg0 = false$ , otherwise set  $flg0 = true$ ;
        if ( $flg0 = false$  &&  $count\_i0 < s$ )
            //count_i0 is the number of 0-type ALERT messages for  $v_i$ .
             $v_t$  sends a 0-type ALERT message for  $v_i$  to  $v_j$ ;
    }
}
else {
    if ( $v_t$  is the node prev in  $R$  ||  $v_t$  is a common neighbor of  $v_i$  and the node prev in  $R$ ) {
        Search the local list RTL $T$  for the record of the node prev sending report  $R$  to  $v_i$ .
        If not found, set  $flg1 = false$ , otherwise set  $flg1 = true$ ;
        if ( $flg1 = false$  &&  $count\_i1 < s$ )
            //count_i1 is the number of 1-type ALERT messages for  $v_i$ .
             $v_t$  sends a 1-type ALERT message for  $v_i$  to  $v_j$ ;
    }
}
}

```

ALGORITHM 1: Processing of a report on its nonspecified receiver (e.g., v_t).

```

/*When  $v_f$  hears an  $x$ -type ALERT message for  $v_i$  reported by  $v_t$  to  $v_j$ .*/
if ( $v_f$  has heard the ALERT message) ignore this ALERT message;
else if ( $v_f$  is exactly  $v_j$ ) {
    if ( $v_t$  is not a neighbor of  $v_i$ ) discard this ALERT message;
    else {
        switch ( $x$ ) {
            case 0:  $count\_i0++$ ; break;
            case 1:  $count\_i1++$ ; break;
        }
    }
}
else if (it is a broadcast ALERT message) {
    //The ALERT message needs to be forwarded.
    if ( $v_f$  is a neighbor of  $v_j$  &&  $v_f$  does not hear any node unicast this ALERT message to  $v_j$ )
         $v_f$  unicasts this ALERT message to  $v_j$ ;
}
}

```

ALGORITHM 2: Processing of ALERT message.

```

/* $v_j$  is processing the report  $R$  sent by  $v_i$ .*/
if ( $v_i$  is the CoS node in  $R$ ) {
    if ( $count\_i0 \geq s$ ) return 1;
}
else { //  $v_i$  is a forwarding node.
    if ( $count\_i1 \geq s$ ) return 1;
}
return 0;

```

ALGORITHM 3: Processing of T_a 's expiration.

For convenience of reference, the collusion attack by fraudulently using neighbors' information is denoted as the CAUNI attack.

(3) Collusion attack by acting as a forwarding node.

The process of the malicious compromised node v_i forging an event report is similar to that in the CAUNI attack. However, in order to pass the verification of the next-hop forwarding node, v_i does not use itself as the CoS node but selects another endorsing node as the CoS node and sets the prev in the forged report as the ID of one of its neighbors. Then, v_i pretends to be a forwarding node to send the forged report to the next hop. For convenience of reference, the collusion attack by acting as a forwarding node is denoted as the CAAFN attack.

(4) Antireporting collusion attack by acting as a forwarding node.

The process of the malicious compromised node v_i forging an event report is similar to that in the CAAFN attack. The difference is that in order to avoid being reported by the surrounding nodes, v_i sets the prev in the forged event report as the ID of a node more than 2 hops away. For convenience of reference, the antireporting collusion attack by acting as a forwarding node is denoted as the ACAAFN attack.

Next, the security features of SEF [5], GFFS [8], EGEFS [9], and EMAS will be compared based on the above four types of false data injection attacks, the selective forwarding attack, and the report disruption attack.

SEF, GFFS, EGEFS, and EMAS can resist the CAFL attack to a certain extent. SEF can filter out some false reports by verifying the MACs. In addition to the MAC verification, GFFS also verifies the legitimacy of the endorsing nodes' locations, which will filter out some false reports that have passed the MAC verification. Therefore, GFFS is more resistant to the CAFL attack than SEF. Besides the MAC verification and the verification of legitimacy of the endorsing nodes' locations, EGEFS also verifies the authenticity of the endorsing nodes' locations, thereby filtering out every false report. By verifying the MACs and the endorsing nodes' IDs, EMAS can also filter out each false report. Therefore, EGEFS and EMAS are more resistant to the CAFL attack than SEF and GFFS.

SEF, GFFS, EGEFS, and EMAS can resist the CAUNI attack to a certain extent. SEF can filter out some false reports through the MAC verification. GFFS also verifies the legitimacy of the endorsing nodes' locations in addition to the MAC verification. However, under the CAUNI attack, the malicious compromised node fraudulently uses the IDs, locations, and key partitions of its neighbors to forge the endorsements; therefore, most false reports can pass the verification of legitimacy of the endorsing nodes' locations and key partitions in GFFS. As a result, the capability of GFFS to resist the CAUNI attack is only slightly stronger

than that of SEF. EGEFS can filter out all false reports through the MAC verification and the verification of authenticity of the endorsing nodes' locations. EMAS can also filter out all false reports through the MAC verification and the monitoring and reporting mechanism. Therefore, EGEFS and EMAS are more resistant to the CAUNI attack than SEF and GFFS.

The capabilities of SEF and GFFS to resist the CAAFN attack are similar to their capabilities to resist the CAUNI attack, i.e., GFFS's capability to resist the CAAFN attack is only slightly stronger than that of SEF. EGEFS performs the MAC verification, as well as the legitimacy and authenticity verification of the endorsing nodes' locations; however, the authenticity verification of the endorsing nodes' locations is performed only by the 1-hop forwarding node of the CoS node; therefore, some false reports may escape the authenticity verification of the endorsing nodes' locations in EGEFS. Nevertheless, EGEFS is still more resistant to the CAAFN attack than SEF and GFFS. EMAS can filter out each false report through the MAC verification and the monitoring and reporting mechanism; therefore, EMAS is more resistant to the CAAFN attack than EGEFS, GFFS, and SEF.

The capability of GFFS to resist the ACAAFN attack is slightly stronger than that of SEF, and EGEFS is more resistant to the ACAAFN attack than SEF and GFFS (the specific analysis is similar to the analysis of resisting the CAAFN attack). EMAS can filter out all false reports through the MAC verification, the prev verification, and the monitoring and reporting mechanism. Therefore, EMAS is more resistant to the ACAAFN attack than EGEFS, GFFS, and SEF.

For the selective forwarding attack, none of EGEFS, GFFS, and SEF adopts any measure to resist such attack, whereas EMAS can resist such attack with the proposed report forwarding strategy.

For the report disruption attack, none of EGEFS, GFFS, and SEF provides any measure to combat such attack. In EMAS, the CoS node verifies the endorsements sent by the detecting nodes and discards the false endorsements; therefore, EMAS can resist the report disruption attack. In conclusion, the security features of SEF, GFFS, EGEFS, and EMAS are shown in Table 1.

5.2. Communication Overhead. In this section, the communication overhead of SEF [5], GFFS [8], EGEFS [9], and EMAS will be discussed, including the predistribution communication overhead and the report-filtering communication overhead.

5.2.1. Predistribution Communication Overhead. The predistribution communication overhead refers to the communication overhead incurred by each node for distributing the location, key partition, and other information during the system initialization phase. Suppose that SEF, GFFS, EGEFS, and EMAS adopt the same forwarding node selection method, then the predistribution communication overhead for implementing the forwarding node selection method is the same for all the four algorithms and can therefore be

TABLE 1: Security features.

Attack	Capability to resist the attack
CAFL attack	EMAS = EGEFS > GFFS > SEF
CAUNI attack	EMAS = EGEFS > GFFS > SEF
CAAFN attack	EMAS > EGEFS > GFFS > SEF
ACAAFN attack	EMAS > EGEFS > GFFS > SEF
Selective forwarding attack	EMAS > EGEFS = GFFS = SEF
Report disruption attack	EMAS > EGEFS = GFFS = SEF

ignored. Thus, SEF does not incur any additional predistribution communication overhead. In GFFS, each node needs to predistribute c packets containing its location and key partition. In EGEFS, each node needs to broadcast its location and send its location to the sink. In EMAS, each node needs to broadcast its location, key partition, and the IDs of all its neighbors and also needs to send its location to the sink.

It can be seen that SEF has the lowest predistribution communication overhead, and EMAS has higher predistribution communication overhead than EGEFS. Because each node in GFFS needs to broadcast its location and key partition throughout the entire network, GFFS usually has the highest predistribution communication overhead.

5.2.2. Report-Filtering Communication Overhead. The report-filtering communication overhead refers to the energy expenditure for filtering out a false report, which is affected by the report length and the number of hops that the report travels. The longer the report length, the higher the report-filtering communication overhead, and the more hops the report travels, the higher the report-filtering communication overhead.

For fairness, assume that SEF, GFFS, EGEFS, and EMAS all adopt Bloom filter. Denote the length of node ID, key index, location, the counter value C , a normal report without any extra field, and Bloom filter as L_S , L_k , L_L , L_C , L_r , and L_F , respectively. The length of prev in EMAS is denoted as L_p . Then, the length of a report in SEF, GFFS, EGEFS, and EMAS is $LR_{SEF} = L_r + T \times L_k + L_F$, $LR_{GFFS} = L_r + T \times (L_k + L_S + L_L) + L_F$, $LR_{EGEFS} = L_r + L_C + T \times (L_k + L_S + L_L) + L_F$, and $LR_{EMAS} = L_r + L_C + T \times (L_k + L_S + L_L) + L_F + L_p$, respectively. As an example, if $T = 5$, $L_S = 10$ bits, $L_k = 10$ bits, $L_L = 16$ bits, $L_C = 8$ bits, $L_r = 24$ bytes [5], $L_F = 64$ bits, and $L_p = 9$ bits, then LR_{SEF} is about 39 bytes, LR_{GFFS} is about 54 bytes, LR_{EGEFS} is about 55 bytes, and LR_{EMAS} is about 56 bytes.

Denote the energy consumption of transmitting and receiving one byte as e_t and e_r . Assume that each node has C_n neighbors on average. When a node sends a report, all its neighbors will hear the report and consume energy to receive it. Assume that the number of hops that a false report travels in SEF, GFFS, EGEFS, and EMAS is H_{SEF} , H_{GFFS} , H_{EGEFS} , and H_{EMAS} , respectively. Then, the energy consumption for filtering out a false report in SEF, GFFS, EGEFS, and EMAS is $EC_{SEF} = LR_{SEF} \times (e_t + C_n \times e_r) \times H_{SEF}$, $EC_{GFFS} = LR_{GFFS} \times (e_t + C_n \times e_r) \times H_{GFFS}$, $EC_{EGEFS} = LR_{EGEFS} \times (e_t + C_n \times e_r) \times H_{EGEFS} + EV$, and $EC_{EMAS} = LR_{EMAS} \times (e_t + C_n \times e_r) \times H_{EMAS} + EA$, respectively, where EV is the energy consumption for verifying the authenticity of locations in the report, and EA

is the energy consumption caused by the monitoring nodes sending ALERT messages to the current forwarding node.

Although LR_{EMAS} is larger than LR_{SEF} , LR_{GFFS} , and LR_{EGEFS} , and EMAS has to spend communication overhead for ALERT messages; however, EMAS has stronger capability to resist the attacks than SEF, GFFS, and EGEFS, resulting in H_{EMAS} usually being smaller than H_{SEF} , H_{GFFS} , and H_{EGEFS} . Therefore, EMAS still has advantages in energy expenditure for filtering out a false report in most cases, and the simulation results also verify this conclusion.

5.3. Storage Overhead. Assume that SEF [5], GFFS [8], EGEFS [9], and EMAS all adopt the proposed report forwarding strategy, then the storage overhead for report forwarding strategy can be overlooked because it is almost the same for the four algorithms. Thus, the average storage overhead of each node for implementing the en-route filtering strategy is mainly considered here.

Each node in SEF needs to store k keys and k key indexes, while each node in GFFS needs to store extra c packets $\{S_i, L_i, U_i\}$. In EGEFS, each node needs to store k keys, k key indexes, and locations of its neighbors, whereas in EMAS, each node needs to additionally store the key partition and neighbor IDs of each neighbor.

Let the length of a key, key index, node ID, location, and U_i be L_b , L_k , L_S , L_L , and L_U , respectively. Suppose each node has C_n neighbors on average, then, the average storage overhead of each node in SEF, GFFS, EGEFS, and EMAS is $AS_{SEF} = k \times (L_b + L_k)$, $AS_{GFFS} = k \times (L_b + L_k) + c \times (L_S + L_L + L_U)$, $AS_{EGEFS} = k \times (L_b + L_k) + C_n \times L_L$, and $AS_{EMAS} = k \times (L_b + L_k) + C_n \times (L_L + L_U + C_n \times L_S)$, respectively.

It can be seen that the average storage overhead in SEF is the smallest and that in EGEFS is smaller than that in EMAS. GFFS or EMAS has the highest average storage overhead, depending on the value of the parameter.

6. Performance Evaluation

In this section, the performance of SEF [5], GFFS [8], EGEFS [9], and EMAS will be evaluated from the aspects of average storage overhead of each node, average predistribution energy consumption of each node, and en-route filtering capability against the four types of false data injection attacks. Among them, the metrics for evaluating the en-route filtering capability include the following:

- (1) *en-route filtering probability*, measured as the percentage of false reports dropped by the forwarding nodes;

- (2) *number of traveled hops*, measured as the number of hops that a false report travels; and
- (3) *filtering energy expenditure*, measured as the energy expenditure for filtering out a false report.

6.1. Experimental Environment and Parameter Setting. Based on the WSN simulator explained by Yi et al. [9], extensive simulation experiments were conducted for SEF [5], GFFS [8], EGEFS [9], and EMAS. In the simulation experiments, 250 sensor nodes were randomly deployed in an area of 100 m $\times$ 300 m, and the sink was located at the edge of the area. A global key pool containing 1000 keys was generated, which was divided into 10 key partitions with 100 keys in each partition. Each node randomly stored 50 keys belonging to the same key partition. The report lengths in SEF, GFFS, EGEFS, and EMAS were 39 bytes, 54 bytes, 55 bytes, and 56 bytes, respectively, and the length of ALERT message in EMAS was 7 bytes. The settings for other simulation parameters are shown in Table 2.

The performance of GFFS is affected by c (the number of predistributed packets). In the simulation experiments, the two cases of $c=20$ and $c=40$ were simulated for GFFS, which were denoted as GFFS_20 and GFFS_40, respectively. Furthermore, the performance of these algorithms is also affected by the number of hops between the malicious compromised node and the sink (denoted as MNHop). In the simulation experiments, the two cases of MNHop=5 and MNHop=10 were simulated.

Each simulation experiment was run 1000 times, and one false report was generated in each run. All the experimental results were averaged over 1000 runs.

6.2. Performance of Storage Overhead and Predistribution Energy Consumption. The experimental results of average storage overhead of each node and average predistribution energy consumption of each node are shown in Table 3. When arranged according to the average storage overhead of each node, the order from low to high is SEF, EGEFS, GFFS_20, GFFS_40, and EMAS. When arranged according to the average predistribution energy consumption of each node, the order from low to high is SEF, EGEFS, EMAS, GFFS_20, and GFFS_40. It can be seen that the experimental results are consistent with the conclusions of theoretical analysis.

6.3. Performance of En-Route Filtering. In this section, the performance of en-route filtering under the CAFL attack, the CAUNI attack, the CAAFN attack, and the ACAAFN attack will be evaluated for SEF [5], GFFS [8], EGEFS [9], and EMAS.

6.3.1. Algorithm Performance under the CAFL Attack. Figures 2(a) and 2(b), respectively, show the experimental results of how the en-route filtering probability varies with the number of compromised nodes cn when MNHop=5 and MNHop=10 under the CAFL attack. It can be seen that when

TABLE 2: Simulation parameters.

Parameter	Value
Communication range R_c	30 m
Sensing range R_s	15 m
T	5
S	3
Kn	4
Energy consumption of transmitting 1 byte	16.25 μ J
Energy consumption of receiving 1 byte	12.5 μ J

TABLE 3: Storage overhead and predistribution energy consumption.

Scheme	Average predistribution energy consumption of each node (J)	Average storage overhead of each node (byte)
SEF	0	463
GFFS_20	0.1712	535
GFFS_40	0.2607	608
EGEFS	0.0115	502
EMAS	0.0184	977

MNHop=5 and MNHop=10, the en-route filtering probabilities of EMAS and EGEFS are not affected by cn , both of which are 100%, whereas those of SEF, GFFS_20, and GFFS_40 gradually decrease with the increase in cn . When MNHop=5, the en-route filtering probabilities of EMAS and EGEFS are higher than those of SEF, GFFS_20, and GFFS_40 and that of GFFS_40 is higher than those of SEF and GFFS_20, while SEF has the lowest en-route filtering probability. When MNHop=10, the en-route filtering probability of GFFS_40 is close to 100%, which is similar to those of EMAS and EGEFS and higher than those of SEF and GFFS_20, while SEF has the lowest en-route filtering probability.

Figures 3(a) and 3(b), respectively, show the experimental results of how the number of traveled hops varies with cn when MNHop=5 and MNHop=10 under the CAFL attack. When MNHop=5 and MNHop=10, the numbers of traveled hops of EMAS and EGEFS are not affected by cn , both are 1, which means that both EMAS and EGEFS can filter out false reports within 1 hop; the numbers of traveled hops of SEF, GFFS_20, and GFFS_40 gradually increase with the increase in cn and that of GFFS_40 is smaller than those of GFFS_20 and SEF, while SEF has the largest number of traveled hops.

Figures 4(a) and 4(b), respectively, show the experimental results of how the filtering energy expenditure varies with cn when MNHop=5 and MNHop=10 under the CAFL attack. It can be seen that when MNHop=5 and MNHop=10, the filtering energy expenditures of all the algorithms gradually increase with the increase in cn . Among them, the filtering energy expenditure of SEF shows the largest increase, whereas that of EMAS shows the smallest increase. When MNHop=5, the filtering energy expenditures are in the following order from low to high: EMAS, GFFS_40, EGEFS, GFFS_20, and SEF, whereas when MNHop=10, the order is EMAS, EGEFS, GFFS_40, GFFS_20, and SEF.

FIGURE 2: En-route filtering probability under the CAFL attack. (a) MNHop = 5. (b) MNHop = 10.

FIGURE 3: Number of traveled hops under the CAFL attack. (a) MNHop = 5. (b) MNHop = 10.

By combining the above experimental results, it can be seen that under the CAFL attack, the en-route filtering capabilities of these algorithms are arranged in descending order as EMAS, EGEFS, GFFS_40, GFFS_20, and SEF.

6.3.2. Algorithm Performance under the CAUNI Attack.

Figures 5(a) and 5(b), respectively, show the experimental results of how the en-route filtering probability varies with the number of compromised nodes cn when MNHop = 5 and MNHop = 10 under the CAUNI attack. It can be seen that when MNHop = 5 and MNHop = 10, the en-route

filtering probabilities of EMAS and EGEFS are not affected by cn , both are 100%, which are higher than those of SEF, GFFS_20, and GFFS_40; the en-route filtering probabilities of SEF, GFFS_20, and GFFS_40 gradually decrease with the increase in cn and that of GFFS_40 is higher than those of SEF and GFFS_20, while SEF has the lowest en-route filtering probability.

Figures 6(a) and 6(b), respectively, show the experimental results of how the number of traveled hops varies with cn when MNHop = 5 and MNHop = 10 under the CAUNI attack. It can be seen that when MNHop = 5 and MNHop = 10, the numbers of traveled hops of EMAS and

FIGURE 4: Filtering energy expenditure under the CAFL attack. (a) MNHop = 5. (b) MNHop = 10.

FIGURE 5: En-route filtering probability under the CAUNI attack. (a) MNHop = 5. (b) MNHop = 10.

EGEFS are not affected by cn , both are 1, whereas those of SEF, GFFS_20, and GFFS_40 gradually increase with the increase in cn and are all larger than 1. The numbers of traveled hops of EMAS and EGEFS are the smallest, whereas that of SEF are the largest, which means that EMAS and EGEFS have the highest en-route filtering efficiency, whereas SEF has the lowest en-route filtering efficiency.

Figures 7(a) and 7(b), respectively, show the experimental results of how the filtering energy expenditure varies with cn when MNHop=5 and MNHop=10 under the CAUNI attack. It can be seen that when MNHop=5 and MNHop=10, the filtering energy expenditure of EMAS

basically stays the same with the increase in cn , whereas those of EGEFS, GFFS_40, GFFS_20, and SEF gradually increase with the increase in cn . The filtering energy expenditure of EMAS is lower than that of EGEFS under some circumstance (e.g., when $cn < 5$) but higher than that of EGEFS under some other circumstance (e.g., when $cn > 5$). Therefore, EMAS and EGEFS can be considered to have similar performance of filtering energy expenditure. The filtering energy expenditures of EMAS and EGEFS are lower than those of GFFS_40, GFFS_20, and SEF and that of GFFS_40 is slightly lower than that of GFFS_20 but higher than that of SEF.

FIGURE 6: Number of traveled hops under the CAUNI attack. (a) MNHop = 5. (b) MNHop = 10.

FIGURE 7: Filtering energy expenditure under the CAUNI attack. (a) MNHop = 5. (b) MNHop = 10.

By combining the above experimental results, it can be seen that under the CAUNI attack, EMAS and EGEFS have similar en-route filtering capabilities, and the en-route filtering capabilities of these algorithms are arranged in descending order as EMAS and EGEFS, SEF, GFFS_40, and GFFS_20.

6.3.3. Algorithm Performance under the CAAFN Attack. Figures 8(a) and 8(b), respectively, show the experimental results of how the en-route filtering probability varies with the number of compromised nodes cn when MNHop = 5

and MNHop = 10 under the CAAFN attack. It can be seen that when MNHop is 5 or 10, the en-route filtering probability of EMAS is 100%, which is not affected by cn , whereas those of EGEFS, SEF, GFFS_20, and GFFS_40 gradually decrease with the increase in cn . The en-route filtering probabilities of these algorithms are arranged in descending order as EMAS, EGEFS, GFFS_40, GFFS_20, and SEF.

Figures 9(a) and 9(b), respectively, show the experimental results of how the number of traveled hops varies with cn when MNHop = 5 and MNHop = 10 under the CAAFN attack. It can be seen that when MNHop is 5 or 10, the number of traveled hops of EMAS is 1, which is not

FIGURE 8: En-route filtering probability under the CAAFN attack. (a) MNHop = 5. (b) MNHop = 10.

FIGURE 9: Number of traveled hops under the CAAFN attack. (a) MNHop = 5. (b) MNHop = 10.

affected by cn , whereas those of EGEFS, SEF, GFFS_20, and GFFS_40 gradually increase with the increase in cn and are all larger than 1. The numbers of traveled hops of these algorithms are arranged in ascending order as EMAS, EGEFS, GFFS_40, GFFS_20, and SEF.

Figures 10(a) and 10(b), respectively, show the experimental results of how the filtering energy expenditure varies with cn when MNHop = 5 and MNHop = 10 under the CAAFN attack. It can be seen that when MNHop is 5 or 10, the filtering energy expenditure of EMAS gradually declines with the increase in cn , whereas those of EGEFS, GFFS_40, GFFS_20, and SEF gradually increase with the increase in cn . The filtering energy expenditure of GFFS_40 is slightly lower

than that of GFFS_20, both of which are higher than those of EMAS, EGEFS, and SEF. The filtering energy expenditure of EMAS is the lowest in most cases (except when $cn = 1$) and that of EGEFS is lower than that of SEF in most cases. Therefore, the filtering energy expenditures of these algorithms are arranged in ascending order as EMAS, EGEFS, SEF, GFFS_40, and GFFS_20.

By combining the above experimental results, it is clear that under the CAAFN attack, EMAS has the highest en-route filtering capability, followed by EGEFS; GFFS_40 has close en-route filtering capability to GFFS_20 (GFFS_40 is slightly better); although SEF has slightly poorer performance than GFFS_40 and GFFS_20 in the en-route filtering

FIGURE 10: Filtering energy expenditure under the CAAFN attack. (a) MNHop = 5. (b) MNHop = 10.

FIGURE 11: En-route filtering probability under the ACAAFN attack. (a) MNHop = 5. (b) MNHop = 10.

probability and the number of traveled hops, SEF has significantly better performance than GFFS_40 and GFFS_20 on the aspect of filtering energy expenditure. In general, SEF has higher en-route filtering capability than GFFS_40 and GFFS_20. Therefore, the en-route filtering capabilities of these algorithms are arranged in descending order as EMAS, EGEFS, SEF, GFFS_40, and GFFS_20.

6.3.4. Algorithm Performance under the ACAAFN Attack. Figures 11(a) and 11(b), respectively, show the experimental results of how the en-route filtering probability varies with

the number of compromised nodes cn when MNHop = 5 and MNHop = 10 under the ACAAFN attack. When MNHop is 5 or 10, the en-route filtering probability of EMAS is 100%, which is not affected by cn , whereas those of EGEFS, SEF, GFFS_20, and GFFS_40 gradually decrease with the increase in cn . The en-route filtering probabilities of these algorithms are arranged in descending order as EMAS, EGEFS, GFFS_40, GFFS_20, and SEF.

Figures 12(a) and 12(b), respectively, show the experimental results of how the number of traveled hops varies with cn when MNHop = 5 and MNHop = 10 under the ACAAFN attack. When MNHop is 5 or 10, the number of

FIGURE 12: Number of traveled hops under the ACAAFN attack. (a) MNHop=5. (b) MNHop=10.

FIGURE 13: Filtering energy expenditure under the ACAAFN attack. (a) MNHop=5. (b) MNHop=10.

traveled hops of EMAS is 1, which is not affected by cn , whereas those of EGEFS, SEF, GFFS_20, and GFFS_40 gradually increase with the increase in cn . The numbers of traveled hops of these algorithms are arranged in ascending order as EMAS, EGEFS, GFFS_40, GFFS_20, and SEF.

Figures 13(a) and 13(b), respectively, show the experimental results of how the filtering energy expenditure varies with cn when MNHop=5 and MNHop=10 under the ACAAFN attack. When MNHop is 5 or 10, the filtering energy expenditure of EMAS gradually declines with the increase in cn , whereas those of the other algorithms

gradually increase with the increase in cn . By comprehensive consideration, the filtering energy expenditures of these algorithms can be arranged in ascending order as EMAS, EGEFS, SEF, GFFS_40, and GFFS_20.

In conclusion, under the ACAAFN attack, EMAS has the highest en-route filtering capability, followed by EGEFS; GFFS_40 has slightly higher en-route filtering capability than GFFS_20; by comprehensively considering the results of the en-route filtering probability, the number of traveled hops, and the filtering energy expenditure, it is concluded that SEF has higher en-route filtering capability than

GFFS_40 and GFFS_20. Therefore, the en-route filtering capabilities of these algorithms are arranged in descending order as EMAS, EGEFS, SEF, GFFS_40, and GFFS_20.

7. Conclusions

In this paper, an en-route filtering scheme called EMAS is proposed, which can improve the en-route filtering probability and efficiency via the MAC verification, the verification of the endorsing nodes' IDs and locations, the prev verification, and the monitoring and reporting mechanism. Before generating an event report, the CoS node verifies the endorsements provided by the detecting nodes first and discards the illegitimate ones, so as to defend against report disruption attacks. Furthermore, a report forwarding strategy is proposed to resist selective forwarding attacks, which can balance the residual energy of nodes in the network and prolong the network life. Both theoretical analysis and simulation results show that compared with SEF [5], GFFS [8], and EGEFS [9], EMAS has better performance on the aspects of security, en-route filtering probability, en-route filtering efficiency, and filtering energy expenditure in most cases.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The author declares that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (grant numbers 61572263 and 61872197); the General Program of Natural Science Foundation of the Jiangsu Higher Education Institutions of China (grant number 19KJB520007); and the High-level Talent Research Start-Up Project of Jinling Institute of Technology (grant number Jit-b-202023).

References

- [1] J. A. Luis et al., "An efficient wireless sensor network for industrial monitoring and control," *Sensors*, vol. 18, no. 1, pp. 1–15, 2018.
- [2] L. Liu, G. Han, Y. He, and J. Jiang, "Fault-tolerant event region detection on trajectory pattern extraction for industrial wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2072–2080, 2020.
- [3] A. Rajput and V. B. Kumaravelu, "Fuzzy logic-based distributed clustering protocol to improve energy efficiency and stability of wireless smart sensor networks for farmland monitoring systems," *International Journal of Communication Systems*, vol. 33, no. 4, pp. 1–28, 2020.
- [4] A. Farman, "An intelligent healthcare monitoring framework using wearable sensors and social networking data," *Future Generation Computer Systems*, vol. 114, pp. 23–43, 2021.
- [5] 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, 2005.
- [6] X. Yang, "A novel en-route filtering scheme against false data injection attacks in cyber-physical networked systems," in *Proceedings of the 32nd ICDCS*, pp. 92–101, Macau, China, June 2012.
- [7] Z. Liu, "A cluster-based false data filtering scheme in wireless sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 23, no. 1, pp. 21–45, 2014.
- [8] 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, 2014.
- [9] C. Yi, "Efficient geographical information-based en-route filtering scheme in wireless sensor networks," *KSII Transactions on Internet and Information Systems*, vol. 12, no. 9, pp. 4183–4204, 2018.
- [10] A. Kumar and A. R. Pais, "Multi-sink en-route filtering mechanism for wireless sensor networks, communications in computer and information science," in *Proceedings of the SSCC*, pp. 122–133, Manipal, India, November 2017.
- [11] A. Kumar and A. R. Pais, "Deterministic en-route filtering of false reports: a combinatorial design based approach," *IEEE Access*, vol. 6, pp. 74494–74505, 2018.
- [12] A. Kumar, N. Bansal, and A. R. Pais, "A partial key pre-distribution based en-route filtering scheme for wireless sensor networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 1, pp. 1471–1486, 2021.
- [13] A. Kumar and A. R. Pais, "Blockchain based en-route filtering of false data in wireless sensor networks," in *Proceedings of the COMSNETS*, pp. 1–6, Bengaluru, India, January 2019.
- [14] Z. X. Liu and H. F. Liu, "A neighbor information and one-way hash chain based filtering scheme in wireless sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 39, no. 1–4, pp. 285–311, 2017.
- [15] 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, 2013.
- [16] H.-W. Ferng and N. M. Khoa, "On security of wireless sensor networks: a data authentication protocol using digital signature," *Wireless Networks*, vol. 23, no. 4, pp. 1113–1131, 2017.
- [17] 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, no. 3, pp. 1–13, 2017.
- [18] M. K. Shahzad and T. H. Cho, "A network density-adaptive improved CCEF scheme for enhanced network lifetime, energy efficiency, and filtering in WSNs," *Ad Hoc & Sensor Wireless Networks*, vol. 35, no. 1–2, pp. 129–149, 2017.
- [19] C. Yi, G. Yang, H. Dai, L. Liu, and N. Li, "Public key-based authentication and en-route filtering scheme in wireless sensor networks," *Sensors*, vol. 18, no. 11, pp. 3829–3851, 2018.
- [20] M. K. Shahzad and T. H. Cho, "PKSM: pre-key distribution and Sink mobility in commutative cipher based en-route filtering to extend the network lifetime in WSNs," *Ad Hoc & Sensor Wireless Networks*, vol. 42, no. 1–2, pp. 19–33, 2018.
- [21] M. K. Shahzad, "A. E. F.: Adaptive en-route filtering to extend network lifetime in wireless sensor networks," *Sensors*, vol. 19, no. 18, pp. 1–17, 2019.
- [22] M. K. Shahzad, L. Nkenyereye, and S. M. Riazul Islam, "A fuzzy system based approach to extend network lifetime for

- en-route filtering schemes in WSNs,” in *Proceedings of the ICCAE*, pp. 118–121, Brisbane, Australia, February 2019.
- [23] P. Wang and G. Tu, “Localization algorithm of wireless sensor network based on matrix reconstruction,” *Computer Communications*, vol. 154, pp. 216–222, 2020.
 - [24] I. Bergel and Y. Noam, “Lower bound on the localization error in infinite networks with random sensor locations,” *IEEE Transactions on Signal Processing*, vol. 66, no. 5, pp. 1228–1241, 2018.
 - [25] C. Yi, “Switching algorithm with prediction strategy for maximizing lifetime in wireless sensor network,” *International Journal of Distributed Sensor Networks*, vol. 11, pp. 1–12, 2015.