ISSN:0975-9646

An Active En-route Filtering Scheme for Information Reporting in Wireless Sensor Networks

Naresh K[1*], Pradeep Kumar P[1], Sathish Kumar S[2]
[1]Department of Computer Science & Engineering, Vivekananda Institute of Tech & Sciences, JNTU, Hyderabad, AP, INDIA
[2]Department of Computer Science, Indur Institute of Engineering &Tecnology, JNTU, Hyderabad, AP, INDIA

***Abstract*—In wireless sensor networks, adversaries can injectfalse data reports via compromised nodes and launch DoS attacks against legitimate reports. Recently, a number of filtering schemes against false reports have been proposed. However, they either lack strong filtering capacity or cannot support highly dynamic sensor networks very well. Moreover, few of them can deal with DoS attacks simultaneously. In this paper, we propose a dynamic en-route filtering scheme that addresses both false report injection and DoS attacks in wireless sensor networks. In our scheme, each node has a hash chain of authentication keys used to endorse reports;meanwhile, a legitimate report should be authenticated by a certain number of nodes. First, each node disseminates its key to forwarding nodes. Then, after sending reports, the sending nodes disclose their keys, allowing the forwarding nodes to verify their reports. We design the Hill Climbing key dissemination approach that ensures the nodes closer to data sources have stronger filtering-capacity. Moreover, we exploit the broadcast property of wireless communication to defeat DoS attacks and adopt multipath routing-to deal with the topology changes of sensor networks. Simulation results show that compared to existing solutions, our scheme can drop false reports earlier with a lower memory requirement, especially in highly active sensor networks.**

***Index Terms*—Information reporting, en-route filtering scheme, wireless sensor networks**

I. INTRODUCTION

WIRELESS sensor networks consist of a large number of small sensor nodes having limited computation capacity, restricted memory space, limited power resource, and short-range radio communication device. In military applications, sensor nodes may be deployed in hostile environments such as battlefields to monitor the activities of enemy forces. In these scenarios, sensor networks may suffer different types of malicious attacks. One type is called false report injection attacks [24], in which adversaries inject into sensor networks the false data reports containing nonexistent events or faked readings from compromised nodes. These attacks not only cause false alarms at the base station, but also drain out the limited energy of forwarding nodes. Also, the adversaries may launch DoS attacks against legitimate reports. In selective forwardingattacks [15], they may selectively drop legitimate reports, while in report disruption attacks [19], they can intentionally contaminate the authentication information of legitimate reports to make them filtered out by other nodes. Therefore, it is very important to design a dynamic quarantine scheme to filter these attacks or at least mitigate their impact on wireless sensor networks.

Recently, several schemes such as SEF [20], IHA [24], CCEF [18], LBRS [19], and LEDS [15] have been proposed to address
false report injection attacks and/or DoS attacks. However, they all have some limitations. SEF is independent of network topology, but it has limited filtering capacity and cannot prevent
impersonating attacks on legitimate nodes. IHA has a drawback,
that is, it must periodically establish multihop pairwise keys between nodes. Moreover, it asks for a fixed path between the base station and each cluster-head to transmit messages in both directions, which cannot be guaranteed due to the dynamic topology of sensor networks or due to the use of some underlying routing protocol such as GPSR [7]. CCEF also relies on the fixed paths as IHA does and it is even built on top of expensive public-key operations. More severely, it does not support en-route filtering. LBRS and LEDS utilize location-based keys to filter false reports. They both assume that sensor nodes can determine their locations in a short period of time. However, this is not practical, because many localization approaches [2], [5], [11] take quite long and are also vulnerable to malicious attacks [3], [8],[9]. In LBRS, report disruption attacks are simply discussed, but no concrete solution is proposed. LEDS tries to address selective forwarding attacks by allowing a whole cell of nodes to forward one report, however, this incurs high communication overhead**.**

In this paper, we propose a dynamic en-route filtering scheme to address both false report injection attacks and DoS attacks in wireless sensor networks. In our scheme, sensor nodes are organized into clusters. Each legitimate report should be validated by multiple message authentication codes (MACs), which are produced by sensing nodes using their own authentication keys. The authentication keys of each node are created from a hash chain. Before sending reports, nodes disseminate their keys to forwarding nodes using Hill Climbing approach. Then, they send reports in rounds. In each round, every sensing node endorses its reports using a new key and then discloses the key to forwarding nodes. Using the disseminated and disclosed keys, the forwarding nodes can validate the reports. In our scheme, each node can monitor its neighbors by overhearing their broadcast, which prevents the compromised nodes from changing the reports. Report forwarding and key disclosure are repeatedly executed by each forwarding node at every hop, until the reports are dropped or delivered to the base station.

Our scheme has two advantages:

- We design the Hill Climbing approach for key dissemination, which ensures that the nodes closer to clusters hold more authentication keys than those closer to the base station do. This approach not only balances memory requirement among nodes, but also makes false reports dropped as early as possible.

• Multipath routing is adopted when disseminating keys to forwarding nodes, which not only reduces the cost for updating keys in highly dynamic sensor networks, but also mitigates the impact of selective forwarding attacks. Simulation results show that, compared to existing ones, our scheme can drop false reports earlier with a lower memory requirement, especially in the networks whose topologies change frequently.

The rest of the paper is organized as follows. We introduce the related work in Section II and define system model and goals in Section III. Then, we present our scheme in Section IV and evaluate its performance in Section V. Simulation results are discussed in Section VI. Finally, we summarize the pros and cons of our scheme and point out future work in Section VII.

II. RELATED WORK

We first discuss existing filtering schemes, then introduce some routing protocols used in wireless sensor networks. The routing strategies of these protocols affect the way that sensor nodes can exchange and disseminate key information, so they have significant impact on filtering schemes.

A. **Existing Schemes for Filtering False Reports**

Ye et al. proposed a statistical en-route filtering (SEF) scheme [20] based on probabilistic key distribution. In SEF, a global key pool is divided into n partitions, each containing m keys. Every node randomly picks k keys from one partition. When some event occurs, each sensing node (that detects this event) creates a MAC for its report using one of its random keys. The cluster-head aggregates the reports from the sensing nodes and guarantees each aggregated report contains T MACs that are generated using the keys from T different partitions, where T is a predefined security parameter. Given that no more than T-1 nodes can be compromised, each forwarding node can detect a false report with a probability proportional to . The filtering capacity of SEF is independent of the network topology, but constrained by the value of . To increase the filtering capacity, we can reduce the value of ; however, this allows the adversaries to break all partitions more easily. In addition, since the keys are shared by multiple nodes, the compromised nodes can impersonate other nodes and report some forged events that “occur” in other clusters.

Zhu et al. proposed an interleaved hop-by-hop authentication (IHA) scheme [24]. In this scheme, the base station periodically initiates an association process enabling each node to establish
pairwise keys with other nodes that are t+1 hops away, where t is a security threshold. In IHA, each sensing node generates a MAC using one of its multihop pairwise keys, and a legitimate report should contain t+1 distinct MACs. Since each multihop pairwise key is distinct, IHA can tolerate up to compromised nodes in each cluster instead of in the whole network as SEF does. However, IHA requires the existence of a fixed path for transmitting control messages between the base station and every cluster-head, which cannot be guaranteed by some routing protocols such as GPSR [7] and GEAR [21]. Moreover, the high communication overhead incurred by the association process makes IHA unsuitable for the networks whose topologies change frequently.

Yang et al. presented a commutative cipher based en-route filtering (CCEF) scheme [20]. In CCEF, each node is preloaded with a distinct authentication key. When a report is needed, the base station sends a session key to the cluster-head and a witness key to every forwarding node along the path from itself to the cluster-head. The report is appended with multiple MACs generated by sensing nodes and the cluster-head. When the report is delivered to the base station along the same path, each forwarding node can verify the cluster-head’s MAC using the witness key. TheMACs generated by sensing nodes can be verified by the base station only. CCEF has several drawbacks. First, it relies on fixed paths as IHA does. Second, it needs expensive public-key operations to implement commutative ciphers. Third, it can only filter the false reports generated by a malicious node without the session key instead of those generated by a compromised cluster-head or other sensing nodes.

B. **Routing Protocols of Sensor Networks**

Several distributed distance-vector based routing protocols [17] have been designed and implemented in TinyOS [16]. In these protocols, each node periodically broadcasts its routing cost to the sink, e.g., the base station, and builds a routing table according to the information received from its neighbors. Route is selected based on the routing metrics such as hop count or link quality.

GPSR [7] and GEAR [21] are location-aware routing algorithms, which assume that each node is aware of its own location. Route is determined as the neighbor with the shortest distance to the sink. If all neighbors are farther than a node itself, the node uses a right-hand rule to select the route. In GEAR, the energy level of each neighbor is also taken into consideration in route selection. One observation from GPSR/GEAR is that the path between two nodes is not bidirectional, i.e., the reports from node i to j may choose a different path from that chosen by the reports from node j to i.

Braginsky et al. proposed Rumor [1] routing protocol. In Rumor, when a sensing node detects some event, it creates an agent that is actually a message containing the routing information about the event. The agent follows a straight path to leave from the sensing node and is associated with a maximum TTL. Each node passed by the agent learns the route to the event. If the base station is interested in some event, it sends out a query message. The movement pattern of a query message is similar to that of an agent. When a query message is delivered to a node who knows the route to the event, a path between the base station and the sensing node (the event) can be established.

Although we only discussed a few routing protocols, our scheme can take advantage of any routing protocol that is designed for wireless sensor networks instead of only the protocols we discussed.

III. PROBLEM STATEMENT

A. **System Model**

We model the communication region of wireless sensor nodes as a circle area of radius r , which is called the transmission range. We only consider the bidirectional links between neighbor nodes and assume that sensor nodes simply discard or ignore those links that are not bidirectional. Based on these assumptions, we say that two nodes must be the neighbor of each other and can always communicate with each other if the distance between them is no more than r.

Wireless sensor nodes may be deployed into some target field to detect the events occurring within the field. For example, in a military application, they may be deployed to a battlefield to detect the activities of enemy forces.We assume that sensor nodes form a number of clusters after deployment, each containing at least n nodes. In each cluster, one node is randomly selectedas the *cluster-head*. To balance energy consumption, all nodeswithin a cluster take turns to serve as the cluster-head. That means physically there is no difference between a cluster-head- and a normal node because the cluster-head performs the same sensing job as the normal node.

Given that some event occurs, e.g., tank movement, we assume that at least nodes can detect it simultaneously, where is a predefined system parameter. These nodes detecting the event are called *sensing nodes*. They generate and broadcast the *sensing reports* to the cluster-head. The cluster-head is responsible for aggregating these sensing reports into the *aggregated reports* and forwarding these aggregated reports to the *base station* through some forwarding nodes.

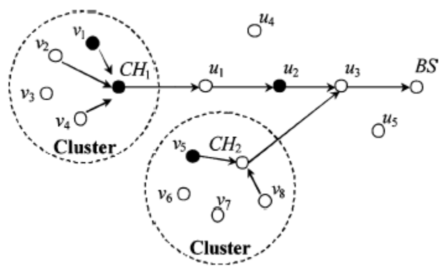


Fig. 1. Sensor nodes are organized into clusters. The big dashed circles outline the regions of clusters. CH and BS denote *Cluster-Head* and *Base Station* respectively.u1~ u5 are forwarding nodes, and u1~u8 are sensing nodes (theycan also serve as forwarding nodes for other clusters). The black dots represent the compromised nodes, which are located either within the clusters or en-route.

Fig. 1 illustrates the organization of sensing nodes in wireless sensor networks. In the figure,CH and BS denote *Cluster-Head* and *Base Station* respectively. $u_1 \sim u_5$are forwarding nodes, and $v_1 \sim v_8$are sensing nodes (they can also serve as the forwarding nodes for other clusters). The black dots represent the compromised nodes, which are located either in the clusters or en-route.

In this paper, we regard data reporting as a task performed at the application layer and ignore the impact of link quality on report delivery. We assume that the protocols at some low layers such as routing layer or MAC layer can handle the failures or collisions in wireless communication by utilizing the mechanisms of acknowledgement and retransmission.

We assume that the topologies of wireless sensor networks change frequently either because sensor nodes are prone to failures or because they need to switch their states between Active and Sleeping for saving energy. Thus, two messages generated by the same cluster may be delivered along different paths to the base station. Moreover, we assume the messages transmitted from a cluster-head to the base station and those from the base station to the cluster-head do not necessarily follow the same path because the underlying routing protocols such as GPSR [7], GEAR [21], or Rumor [1] cannot guarantee this.

B. **Threat Model**

Typically, sensor nodes are not tamper-resistant and can be compromised by adversaries. We assume that each cluster contains at most t-1 compromised nodes, which may collaborate with each other to generate false reports by sharing their secret key information.

In this paper, we consider the following attacks launched by adversaries from the compromised nodes:

• *False report injection attacks*: The compromised nodes can send the false reports containing some forged or nonexistent events "occurring" in their clusters. Moreover, given sufficient secret information, they may even impersonate some uncompromised nodes of other clusters and report the forged events "occurring" within those clusters. These false reports not only cause false alarm at the base station, but also drain out the limited energy of forwarding nodes.

• *DoS attacks*: The compromised nodes can prevent the legitimatereports from being delivered to the base station, by either selectively dropping some reports, (which are called the *selective forwarding attacks* [15]) or intentionally inserting invalid authentication information into the reports to make them filtered by other forwarding nodes (which are called the *report disruption attacks* [19]).

C. Goals

We require that each report be attached with MACs generated by different sensing nodes using their own authentication keys. A false report is defined as one that contains less than valid MACs. Here, selecting different values of gives us a tradeoff between security and overhead. To tolerate more compromised nodes, we can increase the value of , which will incur higher communication overhead because the reports become longer. As we discussed, adversaries can launch false report injection attacks and DoS attacks. Our objective is to design a scheme to detect these attacks or mitigate their impact. Compared to existing ones, our scheme is expected to achieve the following goals:

1) It can offer stronger filtering capacity and drop false reports earlier with an acceptable memory requirement, where the filtering capacity is defined as the average number of hops that a false report can travel.
2) It can address or mitigate the impact of DoS attacks such as report disruption attacks and selective forwarding attacks.
3) It can accommodate highly dynamic sensor networks and should not issue the process of path establishment or reparation frequently.
4) It should not rely on any fixed paths between the base station and cluster-heads to transmit messages.
5) It should prevent the uncompromised nodes from being impersonated. Therefore, when the compromised nodes are detected, the infected clusters can be easily quarantined by the base station.

IV. OUR SCHEME

A. *Overview*

When an event occurs within some cluster, the cluster-head - collects the *sensing reports* from sensing nodes and aggregates them into the *aggregated reports*. Then, it forwards the aggregated reports to the base station through forwarding nodes. Inour scheme, each sensing report contains one MAC that is produced by a sensing node using its authentication key (called *auth-key* for short), while each aggregated report contains distinct MACs, where is the maximum number of compromised
nodes allowed in each cluster.

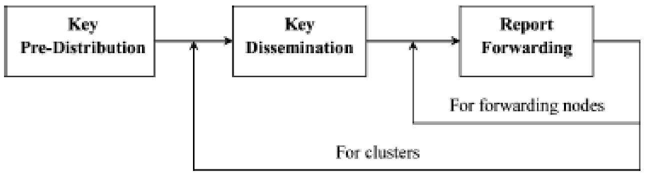


Fig. 2. The relationship between three phases of our scheme. Key predistribution is preformed only once. Key dissemination is executed by clusters periodically. Report forwarding happens at each forwarding node in every round.

In our scheme, each node possesses a sequence of auth-keys that form a hash chain. Before sending the reports, the cluster-head disseminates the first auth-keys of all nodes to the forwarding nodes that are located on multiple paths from the cluster-head to the base station. The reports are organized into rounds, each containing a fixed number of reports. In every round, each sensing node chooses a new auth-key to authenticate its reports. To facilitate verification of the forwarding nodes, the sensing nodes disclose their auth-keys at the end of each round. Meanwhile, to prevent the forwarding nodes from abusing the disclosed keys, a forwarding node can receive the disclosed auth-keys, only after its upstream node overhears that it has already broadcast the reports. Receiving the disclosed keys, each forwarding node verifies the reports, and informs its next-hop node to forward or drop the reports based on the verification result. If the reports are valid, it discloses the keys to its next-hop node after overhearing. The processes of verification, overhearing, and key disclosure are repeated by the forwarding nodes at every hop until the reports are dropped or delivered to the base station.

Specifically, our scheme can be divided into three phases: *key predistribution phase*, *key dissemination phase*, and *report forwarding phase*. In the *key predistribution phase*, each node is preloaded with a distinct seed key from which it can generate a hash chain of its auth-keys. In the *key dissemination phase*, the cluster-head disseminates each node's first auth-key to the forwarding nodes, which will be able to filter false reports later. In the *report forwarding phase*, each forwarding node verifies the reports using the disclosed auth-keys and disseminated ones. If the reports are valid, the forwarding node discloses the auth-keys to its next-hop node after overhearing that node's broadcast. Otherwise, it informs the next-hop node to drop the invalid reports. This process is repeated by every forwarding node until the reports are dropped or delivered to the base station.

Fig. 2 demonstrates the relationship between the three phases of our scheme. *Key predistribution* is performed before the nodes are deployed, e.g., it can be done offline. *Key dissemination* happens before the sensing nodes begin to send the reports. It may be executed periodically depending on how often the topology is changed. Every time the latest (unused) auth-key of sensing nodes will be disseminated. *Report forwarding* occurs at each forwarding node in every round.

B. Detailed Procedure

In the section, we discuss the procedure of each phase in detail.

1) Key Predistribution Phase: Key predistribution needs to be performed only once. It consists of two steps.

Step1: Each node is preloaded with a distinct seed key.

From the seed key, it can generate a sequence of auth-keys using a common hash function h. Thus, each node's authkeys form a hash chain. Let m denote the length of hash chain. Given node v_i well as its seed key , its authkeys $k^{vi}_{m,}$ can be calculated as follows:

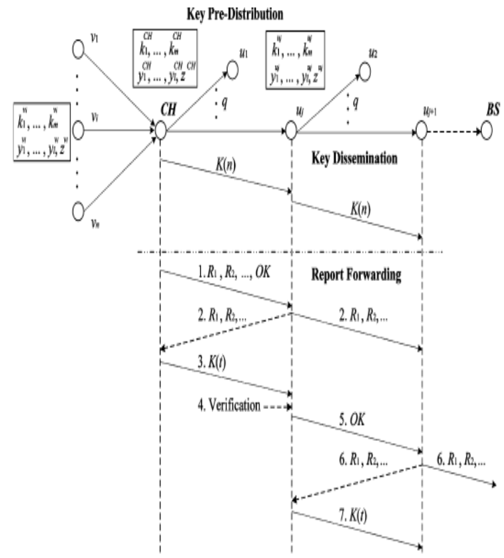


Fig. 3. The detailed procedure of three phases. In the *key predistribution phase*, each node is preloaded with l+1 secret keys y_1……y_l, and z, and can generate a hash chain of auth-keys k_l…..k_m from the seed key k_m. In the *key dissemination phase*, the cluster-head disseminates the auth-keys of all nodes by message K(m) to q downstream neighbor nodes. Every downstream node decrypts some auth-keys from k(n), and further forwards k(n) to q more downstream neighbor nodes, which then repeat the same operation. In the *report forwarding phase*, each forwarding node en-route performs the following steps: 1) It receives the reports from its upstream node. 2) If it receives confirmation message ok, then forwards the reports to its next-hop node. Otherwise, it discards the reports. 3) It receives the disclosed auth-keys within message k(t) and verifies the reports by using the disclosed keys. 4) It informs its next-hop node the verification result.

$$k^{v_i}_{m-1} = h(k^{v_i}_m)$$
$$k^{v_i}_{m-2} = h(k^{v_i}_{m-1}) = h^2(k^{v_i}_m)$$
$$\vdots$$
$$k^{v_i}_1 = h^{m-1}(k^{v_i}_m) \quad (1)$$

where v_i is the node's index h^2 ($K^{vi}{}_m$) , and means hashing $K^{vi}{}_m$ twice. The first key of the chain is $K^{vi}{}_1$, which should also be used the first; meanwhile, it is the last one generated from the seed key. We assume that the base station is aware of each node's seed key, so the adversaries cannot impersonate the uncompromised nodes.

Step2: Besides the seed key, each node is also equipped with l+1 secret keys, where keys (called -keys) are randomly picked from a global key pool (called -key pool) of size , and the rest (called -key) is randomly chosen from another global key pool (-key pool) of size . *Among n nodes of a cluster, we assume that there are at least nodes each having a distinct -key.*

Fig. 3 shows the auth-keys and secret keys possessed by sensor nodes. For example, node 's auth-keys are , and its secret keys are and . If has sufficient memory, it can store all of its auth-keys in memory. Otherwise, it only stores the seed key and generates an auth-key whenever needed.

2) Key Dissemination Phase: In our scheme, the cluster-head discloses the sensing nodes' auth-keys after sending the reports of each round. However, it is vulnerable to such an attack that a malicious node can pretend to be a cluster-head and inject arbitrary reports followed by falsified auth-keys. To prevent this attack, we enforce *key dissemination*, that is, the cluster-head should disseminate the *first* auth-keys of all nodes to the forwarding nodes before sending the reports in the first round. By using the disseminated keys, the forwarding nodes can verify the authenticity of the disclosed auth-keys, which are in turn used to check the validity and integrity of the reports.

Key dissemination should be performed periodically in case that some forwarding nodes aware of the disseminated keys become failed, especially when the network topology is highly dynamic. In this case (of redissemination), the *first unused*, instead of the *first*, auth-keys will be disseminated. The first unused auth-key of a node is called the *current auth-key* of that node. When none of a node's auth-keys has ever been used, the current auth-key is just the first auth-key of its hash chain. The detailed procedure of key dissemination phase is as follows:

Step1: Each node constructs an *Auth* message, which contains l+1 copies of its current auth-key, each encrypted using a different one of its secret keys. For example, given node v_i, its *Auth* message is

$$\begin{aligned} Auth(v_i) = \{\ & v_i,\ j_i,\ id(y_1^{v_i}),\ \{id(y_1^{v_i}), k_{j_i}^{v_i}\}_{y_1^{v_i}}, \\ & \dots,\ id(y_l^{v_i}),\ \{id(y_l^{v_i}), k_{j_i}^{v_i}\}_{y_l^{v_i}}, \\ & id(z^{v_i}),\ \{id(z^{v_i}), k_{j_i}^{v_i}\}_{z^{v_i}}\ \}, \end{aligned} \quad (2)$$

where j_i is the index of its current auth-key. Obviously, $j_i = 1$ for the first dissemination. Here, $id(y1^{vi})$denotes the index of $y1^{vi}$ within the y-key pool, and {.} $y1^{vi}$ means an encryption operation using key . In (2), the index of a secret key is encrypted using the key itself in order to make sure that the decryption is performed using the correct secret key.

Step2: The cluster-head collects the *Auth* messages from all nodes and aggregates them into message K(n)

$$K(n) = \{\ Auth(v_1), \dots, Auth(v_n)\ \} \quad (3)$$

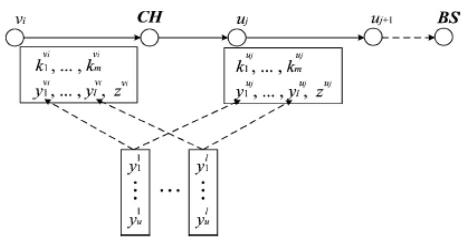
where v_1 …….v_n are the nodes of the cluster.

Step3: The cluster-head chooses q(q > 1)forwarding nodes from its neighbors and forwards them a message,K(n). These q nodes can be selected based on different metrics such as the distance to the base station, the link quality, the amount of energy available, the speed of energy consumption, or a combination of all. How to select an appropriate metric is specific to applications and out of the scope of our paper. In a word, the purpose is to find those nodes that can best forward the reports, so that when some downstream neighbor node dies, the reports can be easily switched to another node without redisseminating K(n).

Step4: When a forwarding node receives K(n), it performs the following operations:

1) It verifies K(n) to see if K(n) contains at least distinct indexes of z-keys. If not, this K(n) is assumed to be forged and should be dropped.
2) It checks the indexes of secret keys in K(n) to see if it has any shared key. When a shared secret key is found, it decrypts the corresponding auth-key using that key and stores the auth-key in its memory. Obviously, it must assure that the decryption key is the correct one by checking the index encrypted along with the authkey.Otherwise, it discards K(n).
3) K(n) does not need to be disseminated to the base station. We define h_{max} as the maximum number of hops K(n) that should be disseminated.Each forwarding node discards the K(n) that has already been disseminated h_{max} hops. Otherwise, it forwards K(n) to other q downstream neighbor nodes, which are selected using the same metric as the cluster-head uses. Each node receiving K(n) repeats these operations, until K(n) gets to the base station or has been disseminated h_{max} hops. Fig. 3 illustrates how a cluster-head disseminates K(n) to its forwarding nodes.

We emphasize that y-keys and z-keys serve for different purposes, although both of them can be used by forwarding nodes to decrypt the auth-keys from K(n). A y-key is mainly used to control how many auth-keys a forwarding node can obtain. However, a z-key is used by its owner to verify the validity of K(n). Since each node has multiple y-keys, the number of y-keys shared by several compromised nodes can easily overwhelm that of distinct y-keys used to endorse each report. Hence,y-keys cannot be used for verification. When the sensing reports are generated continuously and the network topology is highly dynamic, key dissemination should be performed periodically in case that all of q selected downstream nodes die or fail. This period is determined based on the frequency of topology changing. The more often the topology changes, the more often the cluster-head should disseminate the auth-keys. However, we do not discuss how to determine the period because it is out of the scope of this paper.



v_i
CH
u_j
u_{j+1}
BS
$k_1^{v_i}, \dots, k_m^{v_i}$
$y_1^{v_i}, \dots, y_l^{v_i}, z^{v_i}$
$k_1^{u_j}, \dots, k_m^{u_j}$
$y_1^{u_j}, \dots, y_l^{u_j}, z^{u_j}$
y_1^1
y_u^1
y_1^l
y_u^l

Fig. 4. Key predistribution for *Hill Climbing*. Each y-key is randomly selected from a different hash chain of length u=v / l, but the z-keys are still selected from the global key pool.

3) Hill Climbing: We introduce two important observations. First, when multiple clusters disseminate keys at the same time, some forwarding nodes need to store the auth-keys of different clusters. *The nodes closer to the base station need to store more auth-keys than others (typically those closer to clusters) do* because they are usually the hot spots and have to serve more clusters. For example, in Fig. 1,u_3 serves two clusters and u_1 serves only one, so u_3 has to store more auth-keys. Second, *the false reports are mainly filtered by the nodes closer to clusters, while most nodes closer to the base station have no chance to use the auth-keys they stored for filtering*. If we could let the nodes closer to clusters hold more auth-keys, the false reports can be dropped earlier. Therefore, to balance the memory requirement of nodes and provide a higher filtering capacity, we propose *Hill Climbing* approach, which achieves that the nodes closer to clusters hold more auth-keys than those closer to the base station do.

Hill Climbing involves two variations, one for the key predistribution phase and the other for the key dissemination phase.

The first variation is: In Step2 of the key predistribution phase, instead of picking y-keys from a global key pool, each node selects each of its y-keys randomly from an independent hash chain. Specifically, the original y-key pool is partitioned into l equal-sized hash chains, each containing v/l keys that are generated from a distinct seed key. As shown in Fig. 4, the first hash chain contains keys y_1^1 …….y_u^1 , where u=v/l and y_u^1 is the seed key. Similarly, y_1^l …….y_u^l belong to the last chain. Node v_i chooses each of its y-keys $y_1^{v}{}_i$ …….$y_u^{v}{}_i$ from a corresponding chain, and so does node u_j.

It is easy to know that a forwarding node holding a larger index y-key can always decrypt a sensing node's auth-key from K(n) as long as the sensing node's y-key has a smaller index. Inspired by this, we propose the second variation. That is, in Step4 of the key dissemination phase, after a forwarding node decrypts an auth-key from K(n), it updates K(n) by encrypting the auth-key using its own y-key and then forwards the updated K(n) to its downstream neighbor nodes. For example, if K(n) contains $\{id(y_1^{vi}),k_{ji}^{vi}\}_{y1}^{vi}$ [as shown in (2)] $id(y_1^{uj})> id(y_1^{vi})$, u_j and , substitutes $\{id(y_1^{uj}),k_{ji}^{vi}\}_{y1}^{uj}$ for $\{id(y_1^{vi}),k_{ji}^{vi}\}_{y1}^{vi}$ and then forwards the new K(n) to q downstream neighbor nodes. By enforcing this substitution at every forwarding node, the indexes of y-keys contained in K(n) will be increased gradually, just like climbing hill. It becomes harder and harder for the nodes closer to the base station to decrypt the auth-keys from K(n). Consequently, the nodes closer to clusters store more auth-keys, which makes the false reports dropped earlier.

A simpler way to make downstream nodes obtain fewer auth-keys is to discard the auth-keys obtained by forwarding nodes by a gradually increased probability, when these keys approach to the base station. However, *Hill Climbing* has two advantages: 1) It makes the upstream nodes get more auth-keys than not only the downstream nodes but also other upstream nodes at the same positions without using *Hill Climbing*. 2) It eliminates redundant decryptions and verifications because if an auth-key has been decrypted by an upstream node, any downstream node no longer needs to decrypt the key (or use it to verify reports).

4) Report Forwarding Phase: In this phase, sensing nodes generate sensing reports in rounds. Each round contains a fixed number of reports, e.g., 10 reports, where this number is predetermined before nodes are deployed. In each round, every sensing node chooses a new auth-key, i.e., the node's current auth-key, to authenticate its reports. Given node , its sensing report $r(v_i)$ is

$$r(v_i) = \{\, E, v_i, j_i, MAC(E, k_{j_i}^{v_i}) \,\} \qquad (4)$$

where E denotes the event information, j_i is the index of v_i 's current auth-key, and $MAC(E,k_{ji}^{vi})$ is theMAC generated from E using key k_{ji}^{vi} .

In each round, the cluster-head generates the aggregated reports and forwards them to next hop, i.e., one of its q selected downstream forwarding nodes. Then, it discloses the sensing nodes' auth-keys after overhearing the broadcast from the next-hop node. The reports are forwarded hop-by-hop to the base station. At every hop, a forwarding node verifies the validity of reports using the disclosed keys and informs its own next-hop node the verification result. The same procedure is repeated at each forwarding node until the reports are dropped or delivered to the base station.

V. SIMULATION EVALUATION

We study the performance of our scheme by simulation and compare it with others such as SEF, IHA, and CCEF in terms of filtering capacity, fraction of false reports filtered, and memory requirement in different environments.

A. ***Simulation Setup***

- 10^3 nodes are randomly deployed into a 10^3 x 10^3 m^2 square field with the base station located at the center. The transmission range of each node is 50 m. These nodes are divided into 100 clusters, where each cluster contains exactly n=10 nodes.
- Each node picks l=2 y-keys and one z-key, where the size of y-key pool and z-key pool is v=w=20.
- The size of memory used by each node is denoted as mem, and measured by the number of keys that each node stores. Typically, mem=50. In our simulation, each cluster-head disseminates auth-keys to forwarding nodes. One node may need to store the auth-keys from different clusters. It divides its memory into equal-sized slots and assigns one slot to each cluster that it serves.
- Each node forwards K(n) to q=2 selected downstream neighbor nodes, until K(n) reaches the base station or has been forwarded h_{max} hops. Typically, h_{max}=10.
- Each aggregated report contains t=5 MACs, and there are at most t-1=4 compromised nodes in each cluster. The compromised nodes from the same cluster collaborate with each other to share the compromised secret keys.
- To simulate the dynamic topology, we apply a simple ON/OFF operation model, where each node switches its state between ON and OFF periodically. The duration of ON and OFF states satisfies an exponential distribution. We define the percentage of OFF time as *network churn rate*, which indicates the extend of topology changing.
- The routing protocol adopted in our simulation is GPSR. However, IHA and CCEF are not suitable for GPSR because they both require the existence of a fixed path

between each cluster-head and the base station for transmitting control messages in both directions. To make them work on top of GPSR, we revise them accordingly and design a *revised IHA* and a *revised CCEF*. Moreover, in the revised CCEF, we let each forwarding node always keep on forwarding the reports for which it has no witness key. This is different from the original CCEF in which those reports are always discarded.

B. Simulation Results

1) Our scheme drops false reports earlier even with a lower memory requirement. In some scenario, it can drop false reports in 6 hops with only 25 keys stored in each node, but another scheme needs 12 hops even with 50 keys stored.
2) Our scheme can better deal with the dynamic topology of sensor networks. It achieves a higher filtering capacity and filters out more false reports than others in dynamic network.
3) *Hill Climbing* increases the filtering capacity of our scheme greatly and balances the memory requirement among sensor nodes.

VI. CONCLUSION

In this paper, we propose an active en-route quarantine scheme for filtering false data injection attacks and DoS attacks in wireless sensor networks. In our scheme, each node uses its own auth-keys to authenticate their reports and a legitimate report should be endorsed by nodes. The auth-keys of each node form a hash chain and are updated in each round. The cluster-head disseminates the first auth-key of every node to forwarding nodes and then sends the reports followed by disclosed auth-keys. The forwarding nodes verify the authenticity of the disclosed keys by hashing the disseminated keys and then check the integrity and validity of the reports using the disclosed keys. According to the verification results, theyinform the next-hop nodes to either drop or keep on forwarding the reports. This process is repeated by each forwarding node at every hop.

Our scheme has several advantages: 1) Compared with others, our scheme can drop false reports much earlier even with a smaller size of memory. 2) The uncompromised nodes will not be impersonated because each node has its own auth-keys. Therefore, once the compromised nodes are detected, the infected clusters can be easily quarantined. 3) Our *Hill Climbing* key dissemination approach increases filtering capacity greatly and balances the memory requirement among nodes. 4) Each node has multiple downstream nodes that possess the necessary key information and are capable of filtering false reports. This not only makes our scheme adaptive to highly dynamic networks, but also mitigates the impact of selective forwarding attacks. 5) Monitored by its upstream nodes and neighbors, the compromised nodes have no way to contaminate legitimate reports or generate false control messages.

However, to achieve these advantages we have to make some tradeoffs: 1) Our scheme is more complicated than SEF by introducing extra control messages such as K(n), K(t) and ()K . The use of these control message not only increases operation complexity, but also incurs extra overhead, as we already discussed in Section V-B. 2) Like any normal reports, the control messages can also be abused, i.e., they also suffer forgery and DoS attacks. Fortunately, we already proposed some method to prevent the abuse of control messages. As discussed in Section V-D, a forged K(n) can be filtered within whops. 3) The introducing of extra control messages triples the delay of reports. 4) Our scheme requires each node to monitor its downstream nodes and neighbors, which can be achieved by using only bidirectional links. Therefore, sensor nodes have to discard all directed links. 5) In our scheme, each node uses the same auth-key to authenticate all of its reports in the same round. Therefore, this auth-key can only be disclosed after the forwarding nodes forward the reports to their next-hop nodes, which increases memory overhead of the forwarding nodes. 6) Our scheme can not be easily coordinated with other energy-efficient protocols, because in our scheme each node has to be awake until it overhears the broadcast of its next-hop node.

ACKNOWLEDGMENT

The authors appreciate anonymous reviewers for their valuable suggestions and comments.

REFERENCES

[1] D. Braginsky and D. Estrin, “Rumor routing algorithm for sensor networks,” in *Proc. WSNA*, 2002, pp. 22–31.

[2] N. Bulusu, J. Heidemann, and D. Estrin, “GPS-less low cost outdoor localization for very small devices,” *IEEE Personal Commun. Mag.*, vol. 7, no. 5, pp. 28–34, Oct. 2000.

[3] S. Capkun and J. Hubaux, “Secure positioning of wireless devices with application to sensor networks,” in *Proc. IEEE INFOCOM*, 2005, vol.3, pp. 1917–1928.

[4] L. Eschenauer and V. Gligor, “A key-management scheme for distributed sensor networks,” in *Proc. ACM CCS*, 2002, pp. 41–47.

[5] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher, “Rangefree localization schemes in large scale sensor network,” in *Proc. ACM MobiCom*, 2003, pp. 81–95.

[6] C. karlof and D.Wagner, “Secure routing in wireless sensor networks: Attacks and countermeasures,” in *Proc. 1st IEEE Int.Workshop Sensor Netw. Protocols Appl.*, 2003, pp. 113–127.

[7] B. Karp and H. T. Kung, “GPSR: Greedy perimeter stateless routing for wireless networks,” in *Proc. ACM MobiCom*, 2000, pp. 243–254.

[8] L. Lazos and R. Poovendran, “SeRLoc: Secure range-independent localization forWireless sensor networks,” in *Proc. ACMWiSe*, 2004, pp.21–30.

[9] L. Lazos, R. Poovendran, and S. Capkun, “ROPE: Robust position estimation in wireless sensor networks,” in *Proc. IPSN*, 2005, pp. 324–331.

[10] D. Liu and P. Ning, “Establishing pairwise keys in distributed sensor networks,” in *Proc. ACM CCS*, 2003, pp. 52–56.

[11] R. Nagpal, H. Shrobe, and J. Bachrach, “Organizing a global coordinate system from local information on an ad hoc sensor network,” in *Proc. IPSN*, 2003, LNCS 2634, pp. 333–348.

[12] D. Nicolescu and B. Nath, “Ad-hoc positioning systems (APS),” in *Proc. IEEE GLOBECOM*, 2001, vol. 5, pp. 2926–2931.

[13] A. Perrig, R. Szewczyk, V. Wen, D. Culer, and J. Tygar, “SPINS: Security protocols for sensor networks,” in *Proc. ACM MobiCom*, 2001,pp. 189–199.

[14] B. Przydatek, D. Song, and A. Perrig, “SIA: Secure information aggregation in sensor networks,” in *Proc. ACM SenSys*, 2003, pp. 255–265.

[15] K. Ren, W. Lou, and Y. Zhang, “LEDS: Providing location-aware end-to-end data security in wireless sensor networks,” in *Proc. IEEE INFOCOM*, 2006, pp. 1–12.

[16] “TinyOS community forum,” [Online]. Available: http://www. tinyos.net

[17] A.Woo, T. Tong, and D. Culler, “Taming the underlying challenges of reliable multihop routing in sensor networks,” in *Proc. ACM SenSys*, 2003, pp. 14–27.

[18] H. Yang and S. Lu, “Commutative cipher based en-route filtering in wireless sensor networks,” in *Proc. IEEE VTC*, 2004, vol. 2, pp. 1223–1227.

[19] H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbaugh, “Toward resilient security in wireless sensor networks,” in *Proc. ACM MobiHoc*, 2005, pp. 34–45.

[20] F. Ye, H. Luo, S. Lu, and L. Zhang, “Statistical en-route detection and filtering of injected false data in sensor networks,” in *Proc. IEEE INFOCOM*, 2004, vol. 4, pp. 2446–2457.

[21] Y. Yu, R. Govindan, and D. Estrin, “Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks,” Comput. Sci. Dept., Univ. California, Los Angeles, UCLA-CSD TR-01–0023, 2001.

[22] Z. Yu and Y. Guan, “A dynamic en-route scheme for filtering false data injection in wireless sensor networks,” in *Proc. IEEE INFOCOM*, 2006, pp. 1 12.

[23] S. Zhu, S. Setia, and S. Jajodia, “LEAP: Efficient security mechanisms for large-scale distributed sensor networks,” in *Proc. ACM CCS*, 2003, pp. 62–72.

[24] S. Zhu, S. Setia, S. Jajodia, and P. Ning, “An interleaved hop-by-Hop authentication scheme for filtering of injected false data in sensor networks,” in *Proc. IEEE Symp. Security Privacy*, 2004, pp. 259–271.