A Double Key-sharing Based False Data Filtering Scheme in Wireless Sensor Networks

Qian Sun*
School of Information Science and Engineering, Central South University, Changsha, Hunan Province, China, 410083
Email: sun_qian@csu.edu.cn

Min Wu
School of Information Science and Engineering, Central South University, Changsha, Hunan Province, China, 410083
Email: min@csu.edu.cn

***Abstract*—In wireless sensor networks, the attackers can easily inject false data reports from compromising nodes. Previous approaches to filtering false data reports, notably statistical en-route filtering, usually share keys among the nodes in low probabilities, and rely on the forwarding nodes to verify the correctness of the MACs (Message Authentication Codes) carried in each report. Although the results of the notably approach are conspicuous, there still exists several drawbacks. Firstly, compromised nodes from different regions can collaboratively forge false reports such that forwarding nodes cannot detect and filter for not binding the key to their geographical coordinate. Secondly, false reports have to travel several hops before being detected and filtered. In this paper, we propose a Double key-Sharing based false data Filtering scheme (DSF) to cope with those problems. In DSF, nodes are grouped into clusters after deployment, and a blocked region is formed through pair-wise keys closer to the source node. When an event occurs, a legitimate report must carry two types of MACs. In addition, we bind the symmetric keys with the clusters by pre-distributing the key indexes of in-cluster nodes to forwarding nodes. In filtering phase, each forwarding node has to validate not only the correctness of the MACs carried in the report, but also the legitimacy of related locations of all detecting nodes. Moreover, the tail of the data reports can be dropped just outside the blocked region. Extensive analyses and simulations demonstrate that DSF can detect and filter out false reports forged by multiple compromised nodes from different geographical regions, and also outperforms existing schemes in terms of filtering efficiency and energy consumption.**

***Index Terms*—wireless sensor network, double key-sharing, false data filtering, message authentication code**

I. INTRODUCTION

Wireless sensor networks (WSN) are ideal candidates for environment monitoring in various occasions such as military surveillance, habitat monitoring, and health care [1]. In WSN, a large number of sensor nodes with limited resources are usually deployed in a hostile environment. Once a node is compromised, all the keys stored in it will be disclosed by the attacker to easily launch false data injection attacks, i.e., the keys can be abused to inject bogus data reports into sensor networks [2]. Obviously, those attacks may cause not only false alarms but also the depletion of the limited resources of the network.

To protect against false data injection attacks, an en-route false data filtering framework has been proposed [3]. In this framework, every node stores some symmetric keys after being deployed. In case that an event raises, multiple detecting nodes catching the event will simultaneously generate a report that carries t (t >1) distinct MACs (Message Authentication Codes) collectively. Here, t means a security threshold. A MAC is generated by a sensor node using a symmetric key and represents its agreement on the report. As a report being forwarded to the sink over multiple hops, each forwarding node verifies the correctness of the MACs carried in the report probabilistically. A report that carries less than t MACs or a wrong MAC could be marked as invalid and dropped intermediately by nodes or the sink.

Based on the en-route false data filtering framework, several schemes for false data filtering have been proposed in recent years. These schemes have attained good results in protecting the confidentiality, integrity and authenticity of the collected data. However, they suffer from two major drawbacks as follows: First, since any two nodes share symmetric keys in very low probabilities, false data reports injected by the attacker have to travel multiple hops before being filtered out, which results in inefficient energy usage. Second, false data reports forged by multiple compromised nodes from different geographical regions cannot be detected and filtered out, causing the security mechanism to fall completely. For example, as shown in Figure 1, when t=5 and the attacker has compromised nodes S1,…, S5, which makes it possible to forge a report without being filtered out by any forwarding node.

In order to resolve those problems, we propose a Double key-Sharing based false data Filtering scheme (DSF). In DSF, nodes are grouped into clusters and each cluster head establishes relationships with the nodes which lie closer through pair-wise keys to form a blocked region. Furthermore, each pair of nodes shares symmetric

* Corresponding author. Tel.: +86 137 8712 3880

doi:10.4304/jcp.8.2.388-398

keys randomly. We bind the symmetric keys with the clusters by pre-distributing the key indexes of in-cluster nodes to forwarding nodes. When an event occurs, a legitimate report must carry two types of MACs. In filtering phase, each forwarding node has to validate the correctness of these two types of MACs carried in the report, the legitimacy of all detecting nodes' related locations, and drops part of the tail of the reports just outside the blocked region. Extensive analyses and simulations show that DSF outperforms existing schemes in terms of filtering efficiency and energy consumption, and can detect and filter false reports forged by multiple compromised nodes from different geographical region.

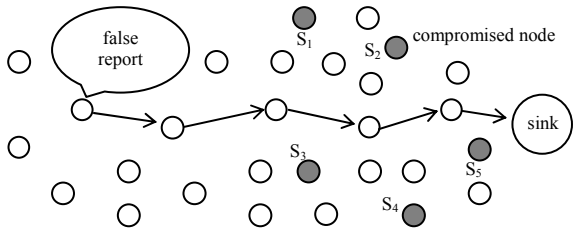

Figure 1. False data injection attack in wireless sensor networks.

The contributions of this paper are as follows: By organizing the sensor nodes into clusters and binding a set of keys to each cluster, false reports forged by compromised nodes from different clusters can be easily detected and filtered out by forwarding nodes. A double-key sharing scheme based en-route filtering scheme is designed. In the scheme, each legitimate report carries two types of MACs. During forwarding, each node in blocked region is able to verify the reports efficiently, while the nodes outside the blocked region only verify reports with some probability. As a result, false reports can be detected and filtered within little hops.

The rest of the paper is organized as follows: The related work is introduced in Section II. The system model is illustrated in Section III. The design of the DSF scheme will be discussed in Section IV. The performance analysis and the simulation results are illustrated in Section V. Section VI concludes this paper.

II. Related Work

The state of the art encrypting techniques can be classified into two categories of symmetric key technique based schemes [3], [4], [5], [6], [7] and asymmetric key based schemes [9], [10], [11], [12]. However, recent research show that symmetric key technique is possibly the only practical approach to establish secure channels among sensor nodes since the low-power sensors have extremely limited computational capability which excludes the applicability of computation-intensive asymmetric key based algorithms [4].

Fan *et al*. [3] first proposed a statistical en-route false report filtering scheme called SEF. In SEF, a key pool is divided into n ($n > t$) partitions, each of which contains m keys. Each node randomly picks k keys from one partition. When an event occurs, a legitimate report is generated by t detecting nodes collaboratively. In the filtering phase, each intermediate node has some probability to verify the correctness of the MACs carried in the report. The sink as the final defense that catches false data reports not filtered out during forwarding. Due to the low filtering probability, false reports have to travel multiple hops before being detected and filtered out in SEF, as illustrated in Figure 1.

Zhu *et al*. [4] proposed an interleaved hop-by-hop filtering scheme called IHA. In IHA, each sensor node is associated with two other nodes on the path to the sink. The pair of associated nodes is called the lower association node and upper association node, respectively. During forwarding, each intermediate node has to check the correctness of the MAC generated by its lower association node. Upon successful verification, it generates a new MAC using the pair-wise key shared with its downstream associated node and replaces the old MAC with the newly generated one. The associations ensure a good filtering capability under determined topology. However, in practical situations, the topology of the sensor network may change frequently due to several reasons, e.g., the sleep of a node, or the move of a node, and thus leading to the decrease of the filtering capacity and the increasing of the maintaining expense.

Yu *et al*. [5] presented a Hill Climbing-based false report filtering scheme named DEFS. In DEFS, sensor nodes are grouped into clusters and each cluster head establishes a path to the sink. Each cluster head distributes keys to the forwarding nodes using Hill Climbing approach through multiple paths. The key distributing method ensures that nodes close to a source cluster hold more keys for the source cluster than those ones farther apart. As a result, the keys stored in each cluster can be balanced, and thus is able to increase the lifetime of the nodes close to the sink. However, transmitting reports using multiple paths is inefficient due to the scarce resource of sensor networks.

Ma *et al*. [6] thought that SEF, IHA and DEFS are limited by the threshold t, and proposed a sink filtering scheme in clusters of heterogeneous sensor networks called RSFS. In addition to basic sensors, RSFS exploits some powerful data gathering sensors termed as cluster heads. Each aggregation report must carry the MACs of all in-cluster nodes. The sink checks the validity of the carried MACs in an aggregation report and filters out the forged data reports. RSFS is able to overcome the limitation of the threshold t, however, all false reports can only be filtered out by the sink and leading to the waste of energy during forwarding.

Yu *et al*. [7] proposed a grouping-based statistical en-route filtering scheme called GRSEF. In GRSEF, nodes are divided into exactly t groups rather than n ($n > t$) groups after being deployed, ensuring that any position in the sensor network can be covered simultaneously by t nodes from distinct groups with high probability. The nodes then fetch keys using a multiple axes-based method. GRSEF is independent of sink stationary and routing protocols, and thus can provide a suitable en-route filtering solution for sensor networks with mobile sinks.

However, multiple axes-based key derivations and maintenance will incur a great cost.

Yang *et al*. [8] proposed a commutative cipher-based false data filtering scheme called CCEF. In CCEF, the source node establishes a secret relationship with the sink, and the intermediate forwarding nodes can then use the witness key to verify the authenticity of the reports without knowing the original session key, and thus can achieve stronger security protection than the existing symmetric key sharing based ways. Wang *et al*. [9] used the elliptic curve cryptography to further improve the security based on CCEF. Ayday et al. [10] exploiting the network coding technique to protect the security of data confidentiality and authenticity. Ren et al. [11] proposed a data authenticity protection approach on particular routings on the research basis of [10].

In all the above schemes, the keys sharing probability is very low in SEF, IHA, DEFS and RSFS. Therefore, false data reports in these schemes have to travel multiple hops before being filtered out, causing energy inefficiency. Furthermore, if the attacker compromised more than t nodes and abuse uses them to forge false reports, then the forwarding nodes are not able to detect and filter. In this paper, we aim to detect and filter out false data reports injected by compromised nodes from different regions and to filter them as early as possible.

III. System Model And Threat Model

We consider a large-scale sensor network in which the sensor nodes will not move after initial deployment. We assume that the sensor nodes are deployed in a high density, so that each stimulus can be detected by multiple sensors in a cluster. The cluster head collects the sensing results by all detecting nodes, and generates a report on behalf of the cluster. Each legitimate report carries the MACs of t members within a cluster. The collected report is finally forwarded toward the sink, usually transmitting a large number of hops.

The cluster based framework is suitable to sensor networks. After being deployed, the node with the smallest ID within its one-hop neighbors is elected as the cluster head, denoted as *CH*. We assume that all the nodes in one-hop cluster can detect the same stimulus happened within one-hop of the *CH* for the reason that the sensing radius of a sensor is much larger than its transmission radius.

The sink has complete knowledge of all the keys, and is equipped with sufficient computation and storage capabilities. False reports with incorrect MACs and locations that sneak through en-route filtering will finally be detected by the sink.

We assume that sensor network has a short safe bootstrapping phase right after network deployment, during which attackers are not able to capture any sensor nodes, and the distributing of location information of the nodes is completed during this time period.

Due to cost constraints, all sensors are not equipped with tamper-resistant hardware. The sink has strong security protection capabilities, and thus cannot be compromised. We assume that the adversary can compromise multiple sensor nodes and take full control of them. Besides false data report injection, a compromised node can also launch various attacks, such as dropping legitimate reports passing through it, replaying legitimate reports and so on. We do not address such attacks in this paper, and only focus on the false data injection attacks [2, 3], in which an attacker injects forged sensing data reports through compromised nodes.

IV. DSF Scheme

DSF includes four phases, namely the keys assignment phase, the report generating phase, the en-route filtering phase and the sink verification phase.

A. Key Assignment Phase

Before deployment, each sensor node is loaded with a unique ID. There is a pre-generated global pool $G = \{K_i: 0 \le i \le N\text{-}1\}$, which is divided into n non-overlapping groups $\{U_i, 0 \le i \le n\text{-}1\}$, each group has m keys. Each node randomly selects one of the partitions and k ($k < m$) keys to store. This type of keys is called R-type keys. Then all sensor nodes are deployed randomly.

CH collects the information of all in-cluster nodes, and generates a Hello message: $\{y, CH, S_1, S_2, \cdots, S_y\}$, where y is a counter with initial value equal to the size of the cluster, and $S_1, S_2, \cdots, S_y$ denotes the in-cluster nodes. Then *CH* forwards the Hello message to the sink and the transmitted hops is controlled by y. Upon receiving the Hello message, the first forwarding node S_i records the last node ID carried in the report and then deletes it. The forwarding node S_i and recorded node S_y are designated as a pair of associated nodes. The node S_i and S_y are called the upstream associated node and downstream associated node respectively. , S_i then inserts its ID into the first fragment of the Hello message, and 1 is subtracted from the counter y. Finally, S_i forwards the message to the next hop. If y is not equal to 0, then all forwarding nodes have to execute the same operations as S_i.

When the last forwarding node S_j receives the Hello message, it generates an ACK message including only the ID itself: $\{S_j\}$. Then S_j transmits the message through the reverse path by which the Hello message is propagated. Each forwarding node inserts its ID into the last fragment of the ACK message. Upon completion, the ACK message will be transmitted to the *CH*, and each in-cluster node records the ID of its upstream associated node from the ACK message.

The associated nodes share pair-wise key based on the algorithms in [15, 16], are called A-type key. Each in-cluster node randomly chooses one of the R-type keys to encrypt its A-type key and ID, and then sends this encrypted information to the *CH*. Finally, *CH* collects all the in-cluster nodes' information and transmits them to the sink. As a result, the adjacent region is then possible to intensively verify the data reports generated by the *CH*. We call this region as the blockade region. The keys assignment procedure can be illustrated as in Figure 2.

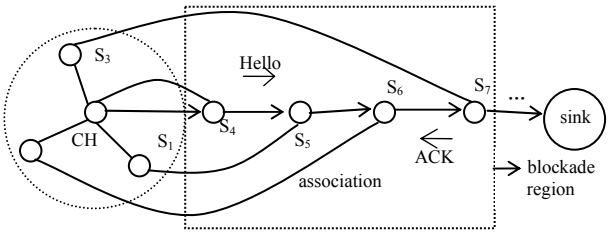

Figure 2. Process of establishing associations and constructing the blockade region

Figure 2 illustrates an example of association establishment. Firstly, the cluster head *CH* generates a Hello message: {*4*, S_1, S_2, S_3, *CH*}. After receiving it, S_4 records *CH* and then deletes it from the message. Inserting S_4 before S_1, it gets a net Hello message {*3*, S_4, S_1, S_2, S_3}. The node *CH* is the downstream association node of S_4. During Hello message forwarding, other nodes operate like S_4. Upon transmitted to S_7, the counter becomes 0, and S_7 generates the initial ACK message: {S_7}. After all the en-route operating as the forwarding nodes, the ACK message becomes {*3*, S_4, S_1, S_2, S_3}. Finally, we get four associated pairs: (*CH*, S_4), (S_1, S_5) , (S_2, S_6) and (S_3, S_7).

B. Report Generationg

When an event occurs, the *CH* broadcasts its reading *e* to all in-cluster members. Upon receiving *e*, a detecting node *S* checks whether *e* is equal to its own sensing value or not. If they match, node *S* randomly selects one R-type key and its A-type key to generate two MACs: M_R, M_A. Node *S* then sends its ID and MACs to *CH*. The *CH* collects the information of *t* in-cluster nodes and generates a report by attaching the IDs, R-type MACs and A-type MACs of *t* detecting nodes after the event *e*. Note that the *t* R-type keys used must be from distinct partitions. The final report sent out by the *CH* to the sink should be as:

$$R_0 : \{C; e; R_1, R_2, \cdots, R_t; M_{R_1}, M_{R_2}, \cdots, M_{R_t}; A_1, A_2, \cdots, A_t; M_{A_1}, M_{A_2}, \cdots, M_{A_t}\} \quad (1)$$

Here *C* is the counter of recording the transmitted hops with initial value equals to the number of nodes in the cluster. R_1, R_2,⋯, R_t denote the R-type keys from the global key pool and M_{R1}, M_{R2}, ⋯ , M_{Rt} indicate the corresponding R-type MACs. A_1, A_2,⋯, A_t are the A-type keys and M_{A1}, M_{A2},⋯, M_{At} mean the corresponding A-type MACs.

We use the Bloom Filters in SEF [14] to map each of these two types of MACs into a string with length *d* bytes: $F=b_0\, b_1 \ldots\, b_{d-1}$, and thus the report becomes:

$$R : \{C; e; R_1, R_2, \cdots, R_t;\ A_1, A_2, \cdots, A_t; F_1; F_2\} \quad (2)$$

The parameter *t* is a tradeoff between the filtering capacity and energy consumption, whose value can be decided by the size of the network and the density. A report including more or less than *t* MACs will be treated as illegitimate by the forwarding nodes. Furthermore, the Bloom Filters uses some system hash values to map the MACs into some shorter string, which can decrease the packet size while at the same time maintain the integrity of the MACs. The detailed introduction of Bloom Filters can be found in [3, 14].

C. En-route Filtering

As a result of the randomized key assignment and the associated pair-wise key sharing, each forwarding node has certain probability to verify the correctness of an R-type MAC or an A-type MAC carried in the report. Moreover, each forwarding node has to drop the tail.

After receiving a report *R*, a forwarding node first checks if *t* R-type MACs and *t* A-type MACs have existed in *R*. If any of them does not meet the requirements, the report would be dropped. Then it checks whether all the R-type keys come from distinct partitions. If any two keys come from the same partition, the report will be discarded. If the node has any of the *t* R-type keys or A-type keys, it reproduces the MAC using its key and compares the result with the corresponding MAC carried in the report. The report will be abandoned if the attached MAC differs from the locally computed one. If they match exactly or the node does not possess any of the keys, *R* will be sent to the next hop. The pseudo code for en-route filtering is given in Figure 3.

/* **Upon received a data report** *R* */
Step1. Checks that *t* {R_k, M_{Rk}} and *t* {A_k, M_{Ak}} tuples exist in *R*; drop *R* otherwise.
Step2. Checks the *t* R-type key indices {R_k, $1 \le k \le t$} belong to *t* distinct partitions; drop *R* otherwise.
Step3. If it has one R-type key $K \in \{K_{Rv}, 1 \le v \le t\}$,it computes $M = K(e)$ and see if the corresponding M_{Rv} is the same as *M*; drop *R* otherwise.
Step4. If it has one A-type key $K \in \{K_{Av}, 1 \le v \le t\}$,it computes $M = K(e)$ and see if the corresponding M_{Av} is the same as *M*; drop *R* otherwise.
Step5. Send *R* to the next hop.

Figure 3. Operations in En-route Filtering

When the counter decreases to 0, it means that the report has passed through the blockade region.

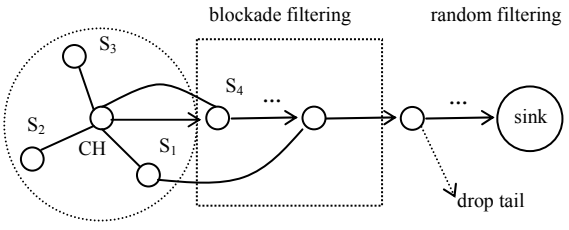

Figure 4. Process of dual filtering and drop tail

As shown in Figure 4, the forwarding nodes do the drop tail operation of the report that just passed through the blockade region, that is to say, to drop the *t* A-type MACs, *t* A-type key indexes and the counter carried in the report. Since this drop tail operation is to drop some

redundant information that only used in the associated verifications, it can obviously save energy.

D. Sink Verification

Because of its complete knowledge of the global key pool and pair-wise keys, the sink serves as the final guard to catches the false data reports not being filtered out by all forwarding nodes. The sink has sufficient capability on computation, storage and security protection. Upon receiving a report, the sink re-computes each of these MACs and compares the results with the attached ones. If any mismatch occurs, the report will be dropped. Any report that forged by t nodes from different clusters will also be detected by the sink. Therefore, DSF can detect bogus reports forged by an attacker with compromised keys in up to t -1 nodes from a cluster. The pseudo code of sink verification is illustrated in Figure 5.

/* **Upon received a data report** R */
Step1. Checks that t $\{R_k, M_{Rk}\}$ and t $\{A_k, M_{Ak}\}$ tuples exist in R; rejects R otherwise.
Step2. Checks the t R-type key indices $\{R_k, 1 \le k \le t\}$ belong to t distinct partitions; rejects R otherwise.
Step3. Computes $M = K(e)$ using the related R-type key and see if the corresponding M_{Rv} is the same as M. If there is a mismatch, R is rejected.
Step4. Computes $M = K(e)$ using the related A-type key and see if the corresponding M_{Av} is the same as M; drop R otherwise. If there is a mismatch, R is rejected. Only those with MACs that are all correct are

Figure 5. Pseudo-code of sink verification

V. Performance Evaluation

In this section, we will begin by discussing the ability to filter false reports injected by compromised nodes from different geographical region, and then analyze the performance to resist more compromised nodes than existing schemes. We then evaluate the effectiveness of en-route filtering and calculate the energy savings as well as to analyze the storage requirement of DSF. Based on the above results, we discuss how to choose appropriate parameters to improve the detecting probability of DSF and reduce the energy consumption. Finally the results of simulation evaluations will be provided.

A. The Ability to Filter Reports Forged by Compromised Nodes From Difference Geographical Region

As illustrated in Figure 6, we assume the monitoring area to be a $\pi \times R_a^2$ circular, in which deployed W nodes with sensing radius R_1. The sensor nodes $S_1, \cdots, S_5$ in Figure 6 are included in distinct clusters, where nodes $S_6, \cdots, S_{10}$ are all included in the same cluster C_1. The area of a cluster is $\pi \times R_c^2$.

Existing en-route false data filtering schemes (SEF, DEFS, GRSEF *et al.*) cannot detect and filter out false data reports injected by compromised nodes from different geographical region. For example, if an attacker uses the compromised nodes $S_1, \cdots, S_5$ in Figure 6 to inject a data report, the intermediate nodes in existing schemes are not able to detect the report.

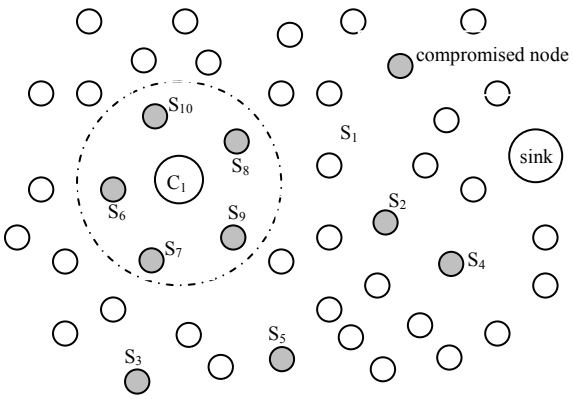

Figure 6. False report forged by compromised nodes from different geographical region in SEF and DSF

Lemma1. *In SEF [3], when each node randomly picks one key partition (totally n key partitions), and if an attacker has compromised N_c sensor nodes in the network ($N_c \ge t$), then the probability for the attacker not to get less than t key partitions can be calculated as,*

$$p_{sef} = \frac{\sum_{j=t}^{n}\left\{C(n,j)\cdot\sum_{k=0}^{j}\left[(-1)^{k+1}\cdot C(j,j-k)\cdot(j-k)^{N_c}\right]\right\}}{n^{N_c}} \quad (3)$$

Proof: Let's first calculate the number of ways of the attacker to get t key partitions out of n key partitions.

Obviously, the number of ways for taking t key partitions out of n key partitions is C_n^t. Furthermore, the number of ways for each of the N_c compromised nodes to pick a key partition out of the selected t partitions (make sure that no key partition is unselected) can be calculated as illustrated in Equation 4.

$$\begin{aligned}\mu &= C(t,t)\cdot t^{N_c} - C(t,t-1)\cdot(t-1)^{N_c} \\ &\quad + C(t,t-2)\cdot(t-2)^{N_c} - C(t,t-3)\cdot(t-3)^{N_c} \\ &\quad + C(t,t-4)\cdot(t-4)^{N_c} - \cdots \\ &\quad + (-1)^{t-1}\cdot C(t,1)\cdot 1^{N_c} \\ &= \sum_{j=0}^{t-1}[(-1)^{j+1}C(t,t-j)(t-j)^{N_c}] \\ &= \sum_{j=0}^{t}[(-1)^{j+1}C(t,t-j)(t-j)^{N_c}]\end{aligned} \quad (4)$$

Similarly, the number of ways for the attacker to pick t key partitions out of n key partitions is $\mu \times C_n^t$. As a result, we can also calculate the number of ways in situations that the attacker picking t+1,⋯, n key partitions out of n key partitions.

The number of ways which an attacker picking at least t key partitions out of n key partitions can be computed as illustrated in Equation 5.

$$\sum_{i=t}^{n}\left\{C(n,i)\cdot\sum_{j=0}^{i}\left[(-1)^{j+1}\cdot C(i,i-j)\cdot(i-j)^{N_c-1}\right]\right\} \quad (5)$$

As a result, we calculate the probability of the attacker to pick at least t distinct key partitions out of n key partitions as P_{sef}.

While in DSF, we bind a set of keys to each cluster, and rely on the forwarding nodes verifying if all of the detecting nodes belong to the same source cluster. DSF is able to limit the damage of compromised nodes to the local area, and thus the false data reports forged by compromised nodes from different geographical region could be detected and filtered out by the intermediate nodes during forwarding. Let's take t =5 as an example, assume that the nodes $S_1,\cdots, S_5$ in Figure 6 are the compromised nodes. Consider that the attacker uses $S_1,\cdots, S_5$ to inject a false data report R, as illustrated in Equation 6.

$$R:\{C; e; R_1, R_2, \cdots, R_5; M_{R_1}, M_{R_2}, \cdots, M_{R_5}; A_1, A_2, \cdots, A_5; M_{A_1}, M_{A_2}, \cdots, M_{A_5}\} \quad (6)$$

R will be detected and dropped by the first forwarding node because the first forwarding node will discover the fact that $S_1,\cdots, S_5$ are from different clusters.

Lemma2. *In DSF, if the attacker has compromised N_c sensor nodes in the network ($N_c \geq t$), and each node randomly picks one key partition (totally n key partitions), then the probability of the attacker to fetch at least t compromised nodes within a same cluster (assume to be C_1 in Figure 6) can be calculated as in Equation 7.*

$$p_{dsf} = \sum_{i=t}^{N_c} C(N_c, i) \cdot (\frac{R_c^2}{R_a^2})^i \cdot (1 - \frac{R_c^2}{R_a^2})^{N_c - i} \quad (7)$$

Proof: We hypothesis the area of a cluster is $\pi \times R_c^2$, e.g., cluster C_1 in Figure 6. Let's also first compute the number of ways of the attacker to get t compromised nodes in C_1.

The probability of one node locating at cluster C_1 is $(R_c / R_a)^2$, so the probability of t sensor nodes locating at cluster C_1 can be illustrated as $(R_1/ R_a)^{2t} \times (1 - (R_1/ R_a)^2)^{Nc-t} \times C_{Nc}{}^t$. Next, the probability of the attacker to fetch t +1,$\cdots$, n compromised nodes in cluster C_1 Should be calculated As a result, we compute the probability of the attacker to fetch at least t compromised nodes within a $\pi \times R_c{}^2$ region as p_{dsf}.

We use simulations to further verify the correctness of our logical analysis, as illustrated in Figure 7. Here we assume that each report carries t = 5 A-type MACs, the total number of key partitions in the global key pool is n = 35, and the comparison of the cluster's radius and the monitoring area's radius is R_c / R_a = 2/5. We use 6000 experiments under random topologies as the simulation results.

From Figure 7, we find that the attacker can break down SEF's security design in a high probability under little compromised nodes, while it can only break down DSF's security design in a very low probability under the same number of compromised nodes. In other words, in order to ruin the security mechanism of DSF completely, the attacker has to compromise a large number of compromised nodes. For example, with ten compromised nodes, the probability of the attacker to ruin SEF and DSF is 97.2% and 1.7%, respectively. With the increasing of compromised nodes, the probability of SEF being broken down increases much more quickly than in DSF. For example, with only 27 compromised nodes the attacker can ruin SEF completely, while in the case of using DSF the attacker has to capture more as 180 compromised nodes. It is to see with both logical analysis and simulation results that DSF is much more resilient than SEF.

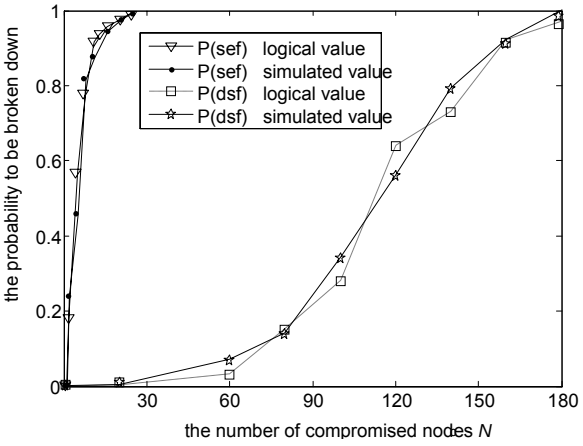

Figure 7. Comparsion between the logical values and simulation values of P(sef) and P(dsf)

B. Filtering Efficiency

Since DSF relies on the t carried R-type MACs and A-type MACs (both are in the form of Bloom Filters) to detect the injected false data reports, an attacker that compromised t or more nodes within a cluster can not only forge reports such that DSF cannot detect, but also launch other kinds of attacks to destroy the network easily. We do not consider such situation here, instead assuming that the number of compromised nodes in a cluster is N_c ($N_c < t$).

To produce a seemingly legitimate data report, the attacker has to fake $t - N_c$ R-type MACs and $t - N_c$ A-type MACs. In SEF [3], the probability for a forwarding node to detect and drop such a faked report is

$$p_1 = k \cdot \frac{t - N_c}{N} \quad (8)$$

While in DSF, the forwarding nodes can not only verify the correctness of the R-type MACs, but also the A-type MACs carried in the reports. The probability of the verification of the A-type MACs is same as p_1. Let the probability of a forwarding node in the blockade region to have one of the corresponding $t - N_c$ R-type keys be p_2.

$$p_2 = \begin{cases} \frac{y - z}{y}, \text{ when } H \leq y \\ \frac{y - z}{H}, \text{ when } H > y \end{cases} \quad (9)$$

The probability of a forwarding node to detect and filter out the forged report is $p_0 = p_1 + p_2 - p_1 p_2$.

$$p_0 = \begin{cases} \dfrac{N(y-z)+k\cdot z(t-N_c)}{N\cdot y}, \text{ when } H \le y \\ \dfrac{N(y-z)+k(t-N_c)(H-y+z)}{N\cdot H}, \text{ when } H > y \end{cases} \quad (10)$$

We evaluate two metrics here, namely the fraction of false reports being detected and dropped before being transmitted to sink, and the hops that a false report which can travel before detected by the intermediate nodes. In DSF, the transmitted hops of a false report can be calculated as

$$\sum_{i=1}^{\infty} i(1-p_0)^{i-1} p_0 = \frac{1}{p_0} \quad (11)$$

Similarly, the number of transmitted hops of a false report in SEF is $1/p_1$. Thus we get the ratio of p_0 and p_1 using Equation 12.

$$\frac{p_0}{p_1} = \frac{p_1 + p_2 - p_1 p_2}{p_1} = 1 + \frac{p_2(1-p_1)}{p_1} \ge 1 \quad (12)$$

Figure 8 illustrates the filtering probability comparison between SEF and DSF according to t and N_c, where H denotes the number of transmitted hops and N represents the number of keys in the global key pool. We supposed that H =20, N =1000, N is divided into n =10 partitions, and each node stores k =50 keys from one randomly selected key partition.

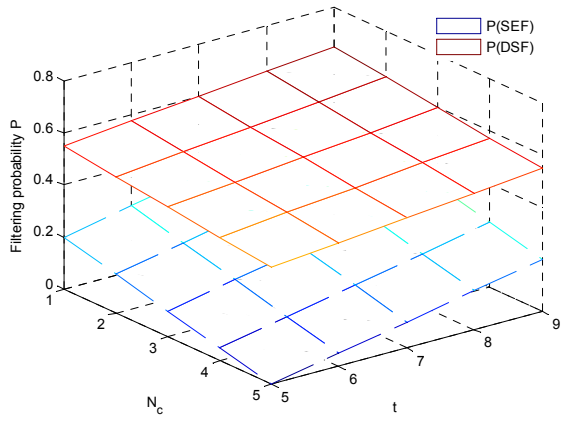

Figure 8. Fluctuation tendency of *p0* and *p1* with t and *Nc* while H =20 hops

As shown in Figure 8, the filtering probability of DSF changes smoothly between 0.5 and 0.7, while the filtering probability of SEF changes sharply between 0 and 0.4. When the number of compromised nodes is small, the difference between p_0 and p_1 is also evident; otherwise, the difference is very distinct. Let's consider an example where N_c =4, p_0 =0.48 and p_1 =0.05 are obtained. So the filtering probability of DSF is obviously higher than SEF.

Figure 9 illustrates the filtering probability of SEF and DSF changing according to the traveled hops H, where N_c =4 and t=5. The others parameters are the same as in Figure 7. From Figure 9 it can be seen that due to the reason that false reports are filtered quickly within the blockade region and the filtering probability of this region is higher than other regions, DSF maintains a high filtering probability reaching larger than 90% at the situation of $H \le y$. As the traveled hops increases, the filtering probability decreases smoothly, and can still larger than 15%. While in SEF, the filtering probability decreases smoothly，while the number of traveled hops increases at a rate of 5%. Therefore, the DSF outperforms SEF in filtering probability.

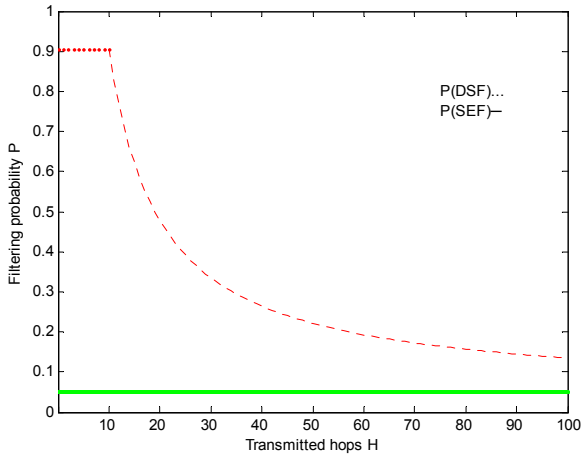

Figure 9. Fluctuation tendency of *p0* and *p1* with H while *Nc* =4 and each report carries t =5 A-type MACs and R-type MACs

C. Energy Consumption

The energy consumption of DSF is mainly posed by four aspects including the communication overhead of each node to distribute the keys to the associated nodes during the bootstrapping phase, the communication energy consumption between the cluster head and other in-cluster nodes when generating a data report, the MAC computation overhead of the forwarding nodes during the en-route verification phase, and the energy consumption of transmitting the reports.

As the researchers in [4] point out that the energy consumption of MAC computation is much smaller than report transmissions. Moreover, the Hello message and ACK message transmitted between the nodes is small, and the packets transmitted between the detecting nodes and the *CH* consumes little energy as well. Furthermore, the distribution of A-type keys during bootstrapping also consumes little energy. Therefore, only the transmission energy consumption of the reports are taken into account.

Compared to SEF, DSF add a counter, t A-type key indexes, t R-type key indexes and two Bloom Filters after each report. These extra fields will cause more energy consumption. However, DSF can save more energy than SEF through the early filtering of the false reports.

As in SEF, DEFS and PVFS, we use the following model to calculate the energy consumption of DSF. Let the size of a normal report without any extra field be the size of the node ID, the length of counter, the length of Bloom Filters and the size of key index be L_r, L_n, L_k, L_u and L_c, combined. Let the length of a SEF packet be L_{SEF}, and the length of a SEF packet be L_{DSF}. Assume that $\alpha_1 = L_{SEF} / L_r = 1 + L_s / L_r + c \times t$, $\alpha_2 = L_{DSF} / L_r = 1 + 2L_s / L_r + L_c /$

L_r +2 × c × t, where c =L_k/ L_r. Let the number of hops that a report has to travel be H, the amount of legitimate reports and false injected reports will be 1 and β, respectively. Therefore, the energy consumed by SEF and DSF in transmitting processes, denoted by E_1 and E_2, can be calculated using Equation 13.

$$E_1 = \alpha_1(H + \beta\frac{1-(1-p_1)^H}{p_1}) \quad (13)$$

$$E_2 = \begin{cases} \alpha_1(H + \beta\dfrac{1-(1-p_0)^H}{p_0}), & \text{when } H \le y \\ \alpha_2 y + \alpha_1(H-y) + \alpha_2\beta\dfrac{1-(1-p_0)^y}{p_0} & \\ +\alpha_1\beta(1-p_0)^y\dfrac{1-(1-p_0)^{H-y}}{p_0}, & \text{when } H > y \end{cases} \quad (14)$$

Figure 10 illustrates the energy consumption of both SEF and DSF changing according to t and β, where L_s = 64 bits, L_k =10 bits, L_r =24 bits, L_c =10bits, n =10bits, N_c =4, m =100 and k =50. From Figure 10 we can see that: (1) the energy consumption of DSF increases slowly than SEF according to t and β, and DSF outperforms SEF in energy saving obviously. When β =9, t =5, the energy consumption of SEF is up to 1150 while maintaining a high filtering probability. While N_c increases, the ratio of p_0 and p_1 raises too, which indicates that the DSF can save more energy than SEF by early filtering the false reports. When N_c reaches the minimum, the difference between the energy consumption of SEF and DSF also diminishes dramatically, even though DSF can still save about 30% more energy than SEF.

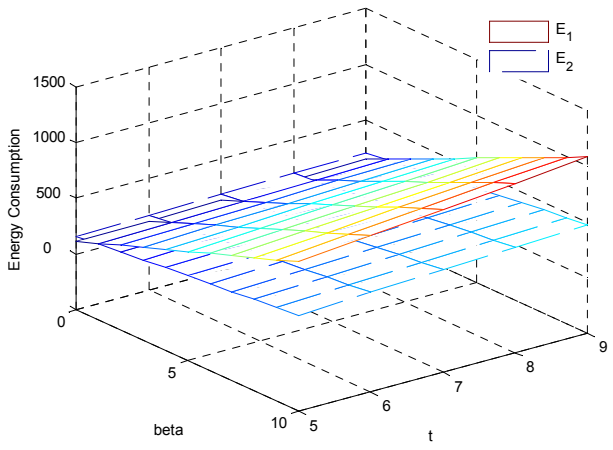

Figure 10. The comparison of the energy consumption of SEF and DSF while H =20 hops and N_c =4

We analyze the energy consumption of SEF and DSF changing according to the traveled hops H, where β =10, t =5, as illustrated in Figure 11. Only when the number of traveled hops is less than 2, the energy consumption of DSF is more than SEF. This is because the report length of DSF is larger than SEF, and only a small number of false reports can be filtered out within little hops. The energy consumption of DSF fluctuates smoothly according to the increasing of traveled hops, while those of SEF changes quickly than DSF with the increasing of traveled hops. The reason for that is DSF can maintain more stable filtering probabilities than SEF and thus saving more energy than SEF.

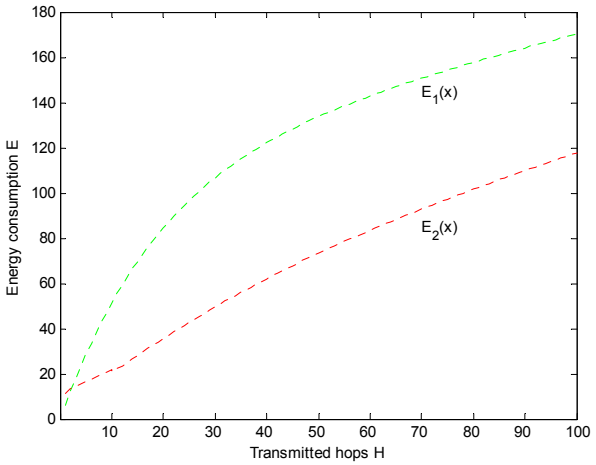

Figure 11. Comparison of Energy consumption between DSF (E_1) and SEF (E_2)

D. Storage Overhead

DSF needs each node to store k +1 keys (k R-type keys and one A-type key). Denote the lengths of a key be 64bits, and then each node needs storage overhead with the size of 88 bytes to store 10 R-type keys and one A-type key. The mainstream nodes, e.g. the MICA2 nodes developed by UCB, being equipped with 3KB SRAM and 128KB ROM can satisfy the storage requirement. Furthermore, as discussed before, we can use the Bloom Filters [14] to map the two types of MACs into two strings in order to further decrease the storage requirement of the sensor nodes. Table 1 illustrates the comparison of the mainstream en-route filtering schemes.

TABLE I.

COMPARISON OF STORAGE OVERHEAD FOR THE MAINSTREAM EN-ROUTE FILTERING SCHEMES

Schemes	Storage overhead
SEF[3]	0.4KB
IHA[4]	4KB
DEFS[5]	3.5KB
GRSEF[7]	2.4KB
DSF	1.2KB

E. Parameters Analysis

Various parameters should be chosen appropriately to make DSF effective. In this section, we mainly discuss the choice of t (the number of MACs each report carries), and N_c (the number of compromised nodes captured by the attacker).

The reasonable value of t is to attain a tradeoff between security and energy consumption. The more extra information each report carries, the more compromised

nodes the attacker needs to fake reports that DSF cannot detect. On the other hand, it also increases the length of the reports and results in more energy consumption. On some low-end nodes, reports cannot be too long to more than 36 bytes. For this reason, t should not be too large. We should choose t appropriately so that it can not only provide sufficient filtering power for en-route, but also it is still small enough to conserve energy.

The number of compromised nodes captured by the attacker mainly affects the security of the filtering scheme. A larger N_c makes it easier for the adversary to fetch two types of keys from a cluster. Therefore, an injected report includes little forged MACs will decrease the filtering efficiency. When N_c is large enough, the attacker can launch various kinds of attacks to destroy the network completely.

F. Simulation Results

In this section, we use simulation experiments to further verify our analysis results. Due to space constraint, we only present results for the ability to resist compromised nodes, filtering probability and energy consumption. The main parameters we considered here including the transmitted hops H and the number of compromised nodes N_c.

Figure 12 shows the network topology structure, where 600 nodes uniformly distribute in a field size of 150× 100m^2. One stationary sink and one stationary source lie in the left side and right side of the region, respectively. The number of hops between the source node and the sink is about 40. The source generates a report every two seconds, and 100 reports in all.

We use a key pool of 1000 keys, divided into 10 partitions, with 100 keys in each partition. Each node has 50 keys in SEF. In DSF, each node stores 50 R-type keys and one A-type key. The transmission radius and sensing radius is 2.5m and 5m, respectively. The packet length of DSF and SEF are the same as in the section III. The power consumptions of transmission and reception in SEF are 6×10^{-3} Joules and 1.2×10^{-3} Joules, respectively. The other parameters are the same as in the section III. The results are averaged over 10 simulated topologies.

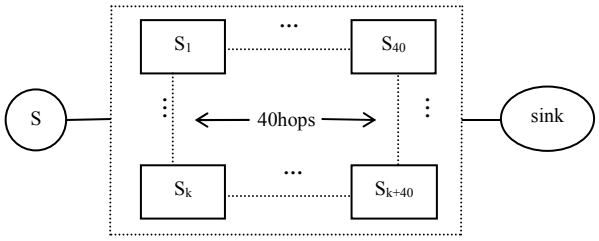

Figure 12. The simulated network topology

In DSF, we assume that the first 10 hops belong to the blockade region. Figure 13 illustrates how the filtering probability of SEF and DSF changes according to the traveled hops H while N_c =4. Figure 13 shows that: (1) In DSF, almost more than 90% false reports can be dropped within the first 10 hops, and the other false reports can also be dropped within 20 hops. While in SEF, only 35% false reports can be dropped within the first 10 hops, whereas half of them can not be filtered out within the first 20 hops. False reports have to travel 35 hops before being dropped without any remainder. E.g., the energy consumption of transmitting 100 false reports in DSF and SEF is 34.56W and 104.04W, respectively. That's to say, DSF can save almost 60% energy than SEF. It is verified that the blockade region in DSF can filter the false reports efficiently.

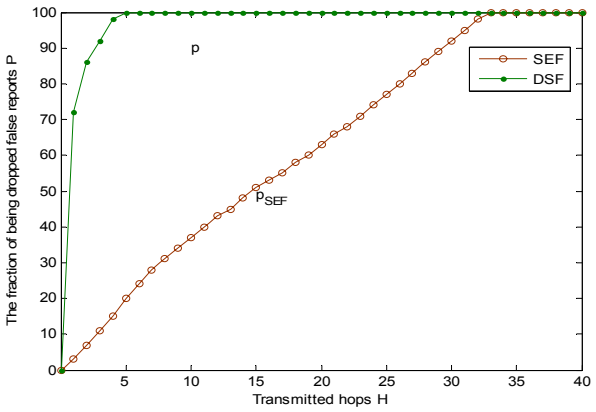

Figure 13. Fluctuation curve of percentage of dropped false reports with H

Figure 14 shows the energy consumption ratio between SEF and DSF according to the changing of H, when β =10, t =5 and N_c =4. From Figure 14 we can see, when H <3, the energy consumption of DSF is less than SEF, and its minimum is equal to 1.5. The curve is like the peaks function with the peak value $H = y$. This is because when the transmitted hops is equal to the associated nodes, the network can filter the false reports with the highest probability, and thus we get the maximum value of E_1/E_2. E.g., when H =3 < y and H =100 > y, the value of E_1/E_2 =1.5, while if $H = y$ =10, the value of E_1/E_2 =2.5. Therefore, under the same transmitting scenario, DSF can save about 30%~60% more energy than SEF.

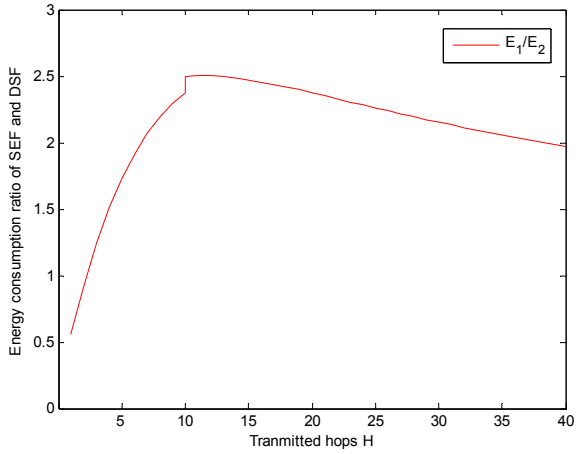

Figure 14. Comparison of the energy consumption ratio between SEF and DSF according to the changing of H

Figure 15 illustrates how the filtering probability of SEF and DSF changes according to the number of

compromised nodes N_c increases. We can see that the filtering probability p in DSF decreases much slowly than in SEF. For example in SEF, we get p =0 when N_c =14, however in DSF, we get p =0 only when N_c =168. It is because DSF verifies the correlations of all detecting nodes on forwarding nodes and thus can resist much more compromised nodes than SEF. This also verifies that the blockade region in DSF can filter the false reports efficiently.

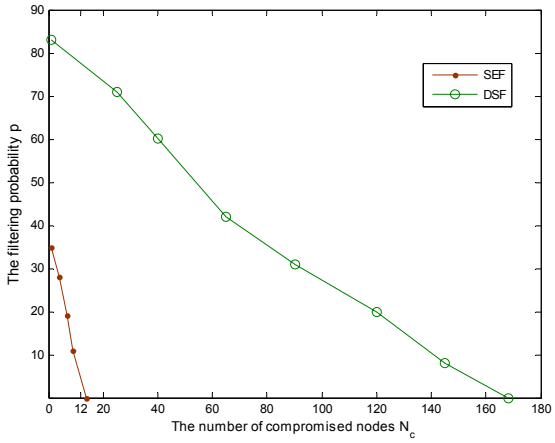

Figure 15. Fluctuation curve of the filtering probability p with the changing of N_c

Figure 16 shows the energy consumption fluctuates with the changing of number of compromised nodes N_c. From Figure 16 we can note that DSF consumes much less energy than SEF. Considering an example of N_c =40, the energy consumption in SEF and DSF is 15.4Joules, 3.2Joules, respectively. This is because when transmitting the same number of hops, DSF can save energy through its earlier filtering of false reports than SEF.

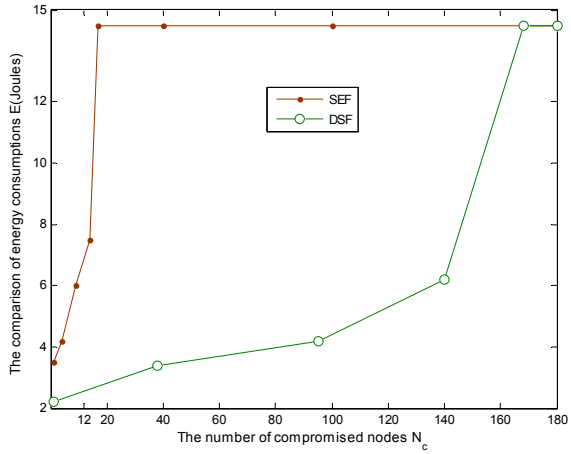

Figure 16. Comparison of energy consumption between SEF and DSF with the changing of number of compromised nodes *Nc*

Moreover, we also can see from both Figure 15 and Figure 16 that, DSF can resist as more as 168 compromised nodes, while SEF can only resist 14 compromised nodes. We deduce that other than SEF, DSF bind the symmetric keys with the clusters by pre-distributing the key indexes of in-cluster nodes to forwarding nodes, and thus can detect and filter out the false reports forged by the compromised nodes from different clusters.

VI. CONCLUSION

Due to the fact that existing false data filtering schemes in wireless sensor networks can not filter false reports efficiently, we propose a Double key-Sharing based false data Filtering scheme (DSF). We adopt the cluster based model to divide the nodes into clusters. Then keys among the nodes are distributed through two ways, namely the random keys sharing and associated keys sharing. The former can further protect the authenticity of the data while the latter is able to guarantee that false reports generated by the cluster can be filtered within only little hops. We further bind the symmetric keys with the source clusters to resist the collaborative false data injection attacks by compromised nodes from different clusters. Moreover, the size of the report becoming less and less as the traveled hops increase for reason that the tail of it is being dropped just outside the blockade region. Extensive analyses and simulations show that DSF outperforms existing schemes in terms of filtering efficiency and energy consumption, and can detect and filter false reports forged by multiple compromised nodes from different geographical region. As for the further work, we would like to optimize the selection of some parameters in order to further improve the performance of DSF.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China under Grant No. 60970096.

REFERENCES

[1] Ren FY, H. HN, C. Lin. “Wireless Sensor Networks,” *Journal of Software*, Vol.14, No.7, pp.1282-1291, 2003.

[2] Z. Su, C. Lin, Feng FJ, Ren FY. “Key Management Schemes and Protocols for Wireless Sensor Networks,” *Journal of Software*, Vol.18, No.5, pp. 1218-1231, 2007.

[3] F. Ye, H. Luo, L. Zhang, “Statistical En-route Filtering of Injected False Data in Sensor Networks,” in *Proceedings of 23th Annual Joint Conference of the IEEE Computer and Communications Societies*, pp.2446~2457, 2004.

[4] S. Zhu, S. Setia, S. Jajodia. “An interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks,” in *Proceeding IEEE symposium on Security and privacy*, pp. 259-271, 2004.

[5] Z. Yu, Y. Guan. “A Dynamic En-route Scheme for Filtering False Data Injection in Wireless Sensor Networks,” in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pp.294-295, 2005.

[6] M. MA. “Resilience of sink filtering scheme in wireless sensor networks,” *Computer Communications*, Vol. 30, No.1, pp.55-65, 2006.

[7] L. Yu, Li JZ. “Grouping-based Resilient Statistical En-route Filtering for Sensor Networks,” in *Proceedings of 28th Annual Joint Conference of the IEEE Computer and Communications Societies*, pp.1782-1790, 2009.

[8] H. Yang, S. Lu. "Commutative Cipher Based En-route filtering in Wireless Sensor Networks," *Vehicular Technology Conference*. Los Angeles, USA, pp.1223-1227, 2004.

[9] H.Wang, Q.Li. "PDF: A Public-key based False Data Filtering Scheme in Sensor Networks," in *Proceedings of the International Conference on Wireless Algorithms, Systems and Applications*, pp.129~138, 2007.

[10] K. Ren, W. Lou, Y. Zhang. "Providing location-aware End-to-end Data Security in Wireless Sensor Networks," in *Proceedings of the IEEE Conference on Computing and Communicating*, pp.585-598, 2006.

[11] Ayday E, Delgosha F, Fekri F. "Location-aware security services for wireless sensor networks using network coding," *IEEE Conference on Computer Communications*. Anchorage, Alaska, USA, pp.226-1234, 2007.

[12] Mao G, Fidan B, Anderson B. "Wireless Sensor Network Localization techniques," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, pp. 2529-2553, 2007.

[13] Bose P, Morin B, Stojmenovic I, Urrutia J. "Routing with guaranteed delivery in ad hoc wireless networks," *Wireless Networks*, Vol.7, No.6, pp. 609-616, 2001.

[14] Peng S. L, Li S. S, Liao X. K, Peng Y. X, Xiao N. "Estimation of a Population Size in Large-Scale Wireless Sensor Networks," *Journal of Computer science and technology*, Vol. 24, No. 5, pp.987-996, 2009.

[15] B. Bloom, "Space/Time trade-offs in hash coding with allowable errors," *Communications of the ACM*, Vol. 13, No. 7, pp. 422-426, 1970.

[16] D. Liu, P. Ning, "Location-Based Pairwise Key Establishments for Static Sensor Networks," in *Proceedings of the 1st ACM workshop on Security in Ad Hoc and Sensor Networks*, pp. 72-82, 2003.

[17] D. Liu, P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," in *Proceedings of the 10th ACM conference on Computer and communications security*, pp.52-61, 2005.

[18] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. M´eard, "Resilient Network Coding in the Presence of Byzantine Adversaries," in *Proceedings of IEEE INFOCOM'07*, 2007.

[19] H. Wang, Q. Li, "Efficient Implementation of Public Key Cryptosystems on Mote Sensors," in *International Conference on Information and Communication Security*, LNCS 4307, Raleigh, NC, pp. 519–528, December 2006.

[20] Wengsheng Zhang, Guohong Cao, "Group rekeying for filtering false data in sensor networks," in *Proceedings of the IEEE Conference on Computing and Communicating*, pp. 503-514, 2005.

[21] P. Traynor, R. Kumar, H. B. Saad, G. Cao, T. L. Porta, "Liger: Implementing Efficient Hybrid Security Mechanisms for Heterogeneous Sensor Networks," in *Proceeding of the MOBISYS'06*, Uppsala, Sweden, June 2006.

[22] T. Arampatzis, J. Lygeros, and S. Manesis, "A survey of applications of wireless sensors and wireless sensor networks," in *Proceedings of IEEE International Symposium on Intelligent Control*, Limassol, Cyprus: IEEE, Vol.1, pp. 719–724, Jun. 2005.

[23] M. Szydlo. "Merkle tree traversal in log space and time, In Advanced Cryptol," Lecture Notes in *Computer Science*, C. Cachin and J. Camenisch, Berlin: *Springer-Verlag*, Vol. 3027, pp. 541–554, May 2004.

[24] D. J. Malan, M. Welsh, M. D. Smith. "A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography," in *Proceeding of IEEE SECON'04*, Santa Clara, CA, pp. 71–80, October 2004.

[25] R. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, P. Kruus, "Tinypk: Securing sensor networks with public key technology," in *Proceeding of ACM SASN'04*, Washington, DC, pp. 59–64, October 2004.

[26] J. Baek, Y. Zheng, "Identity–based threshold signature from the bilinear pairings," in *Proceedings of International Technology: Coding and Computing*, Las Vegas, NV, pp. 124-128, April 2004.

[27] Perrig, R. Szewczyk, J. Tygar, V.Wen, D. Culler, "SPINS: Security protocols for sensor networks," *ACM Wireless Network*, pp. 521-534, September 2002.

[28] W. Lou, W. Liu, Y. Fang, "SPREAD: Enhancing data confidentiality in mobile ad hoc networks," in *Proceeding of IEEE INFOCOM'04*, Hong Kong, China, pp. 2404–2413, March 2004,.

[29] L. Eschenauer, V. Gligor, "A key-management scheme for distributed sensor networks," in *Proceeding of ACM CCS'02*, Washington, DC, pp. 41–47, November 2002.

Qian Sun received his B.S. and M.S. degree in computer science from the Central South University, Changsha, China in 1994 and 2006, respectively. He is currently pursuing his Ph.D. in the School of Information Science and Engineering at Central South University, Changsha, China. His research interests are in the area of network coding and security in wireless communications.

Min Wu received his B.S. and M.S. degrees in engineering from the Central South University, Changsha, China in 1983 and 1986, respectively. In July, 1986, he joined the staff of the university, where he is currently a dean of the School of Information Science and Engineering. He was a visiting scholar in the Department of Electrical Engineering, Tohoku University, Sendai, Japan, from 1989 to 1990, a visiting research scholar in the Department of Control and Systems Engineering, Tokyo Institute of Technology, Tokyo, Japan, from 1996 to 1999, and a visiting professor with the School of Mechanical, Materials, Manufacturing Engineering and Management, University of Nottingham, Nottingham, U.K., from 2001 to 2002. He received his Ph. D degree in engineering from the Tokyo Institute of Technology, Tokyo, Japan in 1999. In 2006, he became one of Cheung Kong Scholars Professors established by China's Ministry of Education and Mr. Li Ka-shing. So far, more than 290 academic papers were published, 80 of which were included by SCI and 210 of which were included by EI. His current research interests include robust control and its applications, process control, and intelligent control.