A Probabilistic Voting-based Filtering Scheme in Wireless Sensor Networks

Feng Li and Jie Wu *
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431

{fli4@, jie@cse.}fau.edu

ABSTRACT

In this paper, we study the fabricated report with false votes attack and the false votes on real reports attack in wireless sensor networks. Since most of the existing works addresses the first attack while leaving an easy way for the attackers to launch the second attack, we propose a probabilistic voting-based filtering scheme (PVFS) to deal with both of them simultaneously. On the basis of the en-route filtering scheme, PVFS combines cluster-based organization, probabilistic key assignment, and voting methods. Through both analysis and simulation, we demonstrate that PVFS could achieve strong protection against both attacks while maintaining a sufficiently high filtering power.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols, Wireless Communications

General Terms

Algorithms, Design, Security, Performance

Keywords

Wireless sensor networks, node compromise, probabilistic key assignment, voting method

1. INTRODUCTION

One core functionality of wireless sensor networks (WSNs) is to detect and report events. A WSN is suitable for tasks such as military surveillance, and forest fire monitoring. We usually use sensors to detect events in an unattended or even hostile environment, in which the adversary may capture and compromise several sensors and launch insider attacks.

One such attack is the fabricated report attack, which means compromised nodes can pretend to have detected a nearby event or forward a report supposedly originating from a remote location. If no security counter-measure is provided, our adversary may claim non-existent events happening at an arbitrary location. This kind of attack will not only cause false alarms that waste real-world response efforts such as sending response teams to the event location, but also drains the finite amount of energy in a battery-powered WSN. Detecting and purging fabricated reports injected by compromised nodes is a great research challenge, since our adversary knows all of the security information of the compromised nodes.

Several recent research efforts [1], [2], and [3] have proposed mechanisms to filter out injected fabricated reports in the forwarding process. Their basic idea is: every node is preloaded with some symmetric keys. When an event occurs, multiple surrounding sensors collectively generate a report that carries multiple message authentication codes (MACs). A MAC is generated by a node using one of its symmetric keys and represents its agreement on the report. As a report is forwarded towards the sink over multiple hops, each forwarding node verifies the correctness of the MACs carried in the report probabilistically. A report with an inadequate number of MACs will not be delivered. Once an incorrect MAC is detected, that report is dropped.

These mechanisms offer a base to solve the fabricated report with false votes attack and keep improving the efficiency. But they also facilitate the attackers to launch the false votes on real reports attack. False votes on real reports attack means that the attacker may inject a false MAC for every real report. If the methods in [1], [2], or [3] are used, all these true event reports will be dropped during the routing process.

In this paper, we design a scheme that could address these two types of attacks simultaneously, and maintain the filtering power at a sufficiently high level. To this end, we exploit a voting method together with a cluster-based organization and a probabilistic key assignment. We break WSNs into clusters, bind a set of keys to each cluster, and use a designed probability to select intermediate cluster-heads as verification nodes. A verification node will not drop a report immediately after it finds a false vote, instead it will record the result of current verification. Only when the number of verified false votes reaches a designed threshold will a report be dropped. The vote in this paper has a similar format as the MAC.

The contributions of this paper are as follows:

1. We use a cluster-based organization in our design. By doing so, compromised nodes from different clusters can not collude. It limits the scope that compromised nodes can abuse their keys.

2. We propose a probabilistic key assignment. On average, the cluster-head stores less verification keys than the verification

*This work was supported in part by NSF grants ANI 0073736, EIA 0130806, CCR 0329741, CNS 0422762, CNS 0434533, and CNS 0531410.

IWCMC'06, July 3–6, 2006, Vancouver, British Columbia, Canada.

nodes in other methods. The false votes tend to be detected in the first few steps.

3. We present a voting method. The decision on whether to drop a report depends on whether the recorded result reaches a designed threshold. By adjusting the threshold, we could offer protection for both attacks and accommodate different application scenarios.

The remainder of this paper is organized as follows. Section 2 discusses several related solutions and the general en-route filtering framework. Section 3 introduces the detailed design of the PVFS scheme. We analyze our scheme and present the simulation results in Section 4. Finally, Section 5 concludes this work and outlines future work.

2. BACKGROUND

Node compromise presents severe security threats, as WSNs usually serve mission-critical applications. Three recent proposals, a statistical en-route filtering mechanism (SEF) [1], an interleaved per hop authentication scheme (IHA) [2], and a location-based resilient security solution (LBRS) [3], provide some protection against the fabricated report attack. SEF [1], IHA [2] and LBRS [3] use general en-route filtering framework as the base of their scheme, which is also the base for the PVFS in this paper.

2.1 Related Work

SEF [1] is the first paper that addresses false sensing report detection problems in the presence of compromised sensors. It also presents the general en-route filtering framework, which serves as the base of IHA, LBRS, and PVFS. SEF suffers from the major drawback that if a certain number of nodes, no matter where they locate, have been compromised, the adversary may claim fabricated events at an arbitrary location without the risk of being detected. We use a cluster-based organization where we bind the generation key to the cluster's id in this paper to address this problem.

IHA [2] verifies the reports in a deterministic and hop-by-hop fashion. There are two major drawbacks in IHA. First, the protection breaks down when more than a certain number (over the threshold) of nodes along a path are compromised. Second, it relies on deterministic key sharing. Each node must know one predetermined upstream and one predetermined downstream neighbor and establish symmetric keys with them. IHA presents us the idea to associate nodes along a routing path together and also leads us to use probabilistic key sharing in our design.

In LBRS [3], the terrain is divided into a regular geographic grid, and each cell on the grid is associated with multiple keys. An attacker that has compromised multiple nodes may obtain keys bound to different cells, but he cannot combine such keys to fabricate any event without being detected. LBRS improves the resiliency of en-route filtering, but it requires additional information such as sensor location information.

Although there are obvious innovations in these three mechanisms, all of them make false votes injection attack easier while offering protection against the fabricated report attack. This paper aims to find a way to address both of them, while maintaining a comparatively high filtering power.

2.2 General En-route Filtering Framework

The general en-route filtering framework has three main components: report generation using message authentication codes (we refer to MACs as votes in this paper), en-route filtering, and sink verification.

Table 1: List of notations

N	Number of nodes in the WSN.
N_c	Number of compromised nodes.
C_{id}	Unique cluster ID.
L	Number of keys for one cluster.
n	Number of keys in the global key pool.
K_i	The key with the index i.
s	Required number of votes for a legitimate report.
d_0/d_i	The hops from the original/ i^{th} CH to the sink.
T_t/T_f	Threshold of true/false votes to accept/drop a report.

The general en-route filtering framework requires a legitimate report to carry s (s is the required number of votes a legitimate report should carry) distinct votes, where each vote is generated by a sensor using a symmetric key (the generation key) and represents the node's endorsement on the content of the report. When a real event occurs, multiple detecting nodes jointly generate a complete report with the required s votes and the associated key indices.

The intermediate nodes detect and discard bogus reports injected by compromised nodes. When a node receives a report, it verifies those votes in that report as long as it stores the corresponding verification keys. It follows designed rules (we introduce our rules in Section 3.3) to decide whether to drop a report or further forward it.

Even though the filtering power at each node may be limited, the collective filtering power along the forwarding path can be significant. The more hops a forged report traverses, the higher the chance that it is dropped en-route. Consequently, one can effectively use the resources of WSNs without being hampered by forged reports. The en-route filtering performed by sensor nodes is probabilistic in nature, thus cannot guarantee to detect and drop all forged reports. The sink serves as the final guard in rejecting any forged reports that get through.

3. DESIGN

To achieve better resilience against node capture and offer protection for both the fabricated report with false votes attack and the false votes on real reports attack, we propose a new probabilistic voting-based filtering scheme (PVFS). PVFS takes the general en-route filtering framework as its base and significantly improves it through three mechanisms: cluster-based key organization, probabilistic key assignment, and voting method. Notations are listed in Table 1.

3.1 System Model and Assumptions

We consider a large-scale sensor network in which the nodes do not move after initial deployment. Most nodes in that network behave normally, so we can always collect enough true votes for true reports, and the adversary could not generate enough true votes for a fabricated report. If our adversary has captured most of the sensors, they do not need to launch these two kinds of attacks.

The cluster-based model is naturally suitable for the filtering mechanisms. It breaks the network into clusters. We choose the cluster-based model to organize sensors in PVFS. When sensors are dense enough, with a cluster merge method, we can make each cluster include approximately L nodes. Let $L \geq s$, which guarantees that a real report could collect enough votes. In a cluster, one node is elected to be the cluster-head represented by CH. Each cluster has a unique cluster ID C_{id}. There are many ways to form

clusters, elect CHs, and generate unique cluster IDs [4]. We use a simple cluster formation method in our scheme: a node with the smallest node ID in all its one-hop neighbors is elected as the CH, and its one-hop neighbors join that cluster. The node ID is unique and predetermined before sensor deployment. We assume all the sensors in one-hop cluster are able to detect the same event happening within one-hop of the CH. In most WSNs, the sensing range of a sensor is much larger than its transmission range. So it is reasonable that we make such an assumption. Each CH uses a larger transmission range than other nodes and discovers a route to the sink which only consists of CHs. Each CH could discover c paths to the sink in case of node failure.

We focus on two types of attacks: the fabricated report with false votes attack and the false votes on real reports attack. So the compromised nodes in this model would either fabricate a report and fabricate some false votes to support that report, or cast a false vote on a real report. Compromised nodes may launch several other attacks. For example, it may simply drop every report or modify the report it receives. However, the WSN may employ mechanisms such as Watchdog [10] to solve these attacks. They are out of the scope of this paper. To simplify the discussion, we also assume that at the beginning of sensor deployment, for a very shot period of time, no node is compromised, we could complete cluster formulation, key distribution, and route discovery without been attacked.

3.2 Initialization and Key Assignment

The PVFS requires a pre-generated global key pool of n keys $\mathcal{K} = \{K_i : 0 \leq i \leq n-1\}$. After the sensors have been organized into clusters, we divide the global key pool into cn non-overlapping partitions $\mathcal{K}_{C_{id}} = \{K_i : L \cdot C_{id} \leq i \leq L \cdot (C_{id}+1) - 1\}$. Here L represents the number of keys in each partition and cn is the number of clusters in the WSN, $L \geq s$ and $n = cn \cdot L$.

Each sensor in a cluster selects one key from the partition $\mathcal{K}_{C_{id}}$. The key will be stored in the following format: (i, K_i) where i is the key's index. We can decide a key K_i belongs to which partition by using function $C_{id} \leftarrow \lfloor i/L \rfloor$. So $\mathcal{K}_{C_{id}}$ is bound to the cluster identified by C_{id}. A node will use this key as the generation key to generate a vote for an event report. We can use a method similar to the idea in [5] to complete the key dissemination in each cluster.

Each CH then represents the cluster to select the verification nodes in the upstream CHs. Upstream CHs means those CHs on one or more paths between the CH which starts the selection process (original CH) and the sink. In the route discovery phase, each CH may get all the upstream CHs' IDs and their distance to the sink in hop count, including its own distance to the sink. Let d_0 represent the distance between the original CH and the sink, and d_i represent the distance between the upstream cluster-head CH_i and the sink.

The original cluster will select an intermediate cluster-head CH_i to be a verification node with a probability: $P = d_i/d_0$.

After it has selected a verification node, the original CH will notify one node in its cluster to exchange its generation key with the selected intermediate CH. As we assume that no node will be compromised during the initialization phase, the key could be directly sent to the intermediate CH. However, to relax this assumption, we can use pairwise key establishment protocols such as [6] to establish a session key, and use this session key to securely transmit the generation key to the selected intermediate CH. Each selected intermediate CH will then have a key from the partition $\mathcal{K}_{C_{id}}$ to be its verification key, where the C_{id} is the original cluster's id. Each CH also shares another symmetric key K_{bc} with the sink to generate the signature of its verification decision.

3.3 Report Generation

The voting method is the core of our design. When an event occurs, the clusters nearby compete with each other, and the winner prepares a report. The CH of that cluster generates a report describing the event and broadcasts it in that cluster. Other nodes in that cluster will compare and decide whether the report is consistent with its observation. If so, it casts a vote using its generation key K_i, and the vote should be:

$$Vote : (i, E_{K_i}(H(Report)))$$

After a vote has been generated, the node sends it to the CH. We could utilize methods such as radio resource testing [8] to ensure that each node could cast only one vote for each report to protect our system from the Sybil attack[7], which means a compromised node can present multiple identities.

When CH has received all the votes, it randomly selects a fixed number of votes, including the vote generated by itself, and appends them to the report. That fixed number is s, which is the required vote-set length in our scheme.

$$C_{id}, Report, \{Vote_i\}$$

The CH will then forward the report to its upstream neighbors. Algorithm 1 demonstrates actions of CHs in the report generation phase. Here $N_{C_{id}}$ represent the node set of the cluster identified by C_{id}, R is the report and V is the set that contains the received votes.

Algorithm 1 ReportGen($N_{C_{id}}, R, V, C_{id}, s$)

1: Generate R to describe the event;
2: Broadcast R in the cluster;
3: Wait;
4: **for** $node \in N_{C_{id}}$ **do**
5: Collect one $vote$;
6: Add $vote$ to V;
7: **end for**
8: Select s $vote$s from V, Add to R;
9: Forward R, EXIT;

3.4 En-route Filtering

After a CH receives a report, it will check whether all the votes belong to the same cluster that generated the report, by comparing C_{id} in R with $\lfloor i/L \rfloor$ for each vote in the vote-set. It will then verify the vote if it holds the corresponding verification key and record the verification result using two binary sequences Bin_v and Bin_r. Bin_v and Bin_r record the verified votes and the verified true votes by setting corresponding bits to 1. The length of these two sequences are decided by s. The node will also check if the number of recorded true or false votes has reached the threshold, and decide whether to drop the report or set the accepted flag $Flag_r$ to 1. It will generate a signature of the report and its verification result using its K_{bc} and forward the report if T_f has not been reached. A report in forwarding will be in the form:

$$C_{id}, Report, \{Vote\}, Bin_v, Bin_t, Flag_r, \{SIG\}$$

Algorithm 2 shows actions of intermediate CHs in the report forwarding phase. Here DS_K represents the database of verification keys stores in a CH.

The sink performs final verification on the received reports. It knows all the generation keys, thus it is able to verify every vote in

Algorithm 2 ReportVeri$(R, DS_K, T_f, T_t, K_{bc})$

if $R.Flag_r = 1$ **then**
  Forward R, EXIT;
end if
for each vote (i, K_i) in $R.\{Vote\}$ and $R.Bin_v = 0$ **do**
  if $\lfloor i/L \rfloor \neq C_{id}$ **then**
    $R.Bin_v[i] \leftarrow 1$;
  else if $(i, K_i) \in DS_K$ **then**
    if $D_{k_i}(Vote) = H(R.Report)$ **then**
      $R.Bin_t[i] \leftarrow 1$;
    end if
    $R.Bin_v[i] \leftarrow 1$;
  end if
end for
Count the number of verified true and false votes;
Drop R and EXIT if T_f has been reached;
$R.Flag_r \leftarrow 1$ if T_t has been reached;
Add $E_{K_{bc}}(H(R))$ to $R.\{SIG\}$;
Forward R, EXIT

the report. It can verify those unverified votes, and make the final decision. It will also verify each CH's signature. In this way, the sink serves as the final guard.

3.5 Example

Fig.1 illustrates our idea. All nodes report to the sink S. We consider $L = 9$, $s = 5$ and $T_f = 2$. In the initializing phase, all sensors are organized into clusters. In Fig. 1(a), we use C_0 to represent the cluster with $C_{id} = 0$. CH_0 is the cluster-head of C_0. Each node in the cluster gets a key from the partition $\mathcal{K}_0 = \{K_i : 9 \cdot 0 \leq i \leq 9 \cdot 1 - 1\}$ bound to that cluster. C_0 then picks up its verification nodes. As CH_2 is on the path that connects S and CH_0, $d_1 = 3$ and $d_0 = 4$, CH_2 has a probability $P = \frac{3}{4}$ to be a verification node of C_0. We can compute the probability of CH_3 and CH_4 similarly. Fig. 1 (a) shows that both CH_2 and CH_4 get a verification key of C_0. CH_2 gets the key $(4, key_4)$, and CH_4 gets the key $(7, key_7)$.

When an event occurs near C_0, CH_0 generates a report, and broadcasts it in C_0. If a node finds that the report is in accordance with its observation, it will sign the report. After collecting all the votes, CH_0 randomly selects the votes with $i = 1, 4, 5, 7, 8$ as $s = 5$, and send the report. CH_2 verifies the vote with $i = 4$. As this vote is true, CH_2 sets the corresponding bit in Bin_v and Bin_t to 1 and forward it.

Let us consider another situation. If CH_0 fabricates a report about a non-existing event, it has to fabricate some votes. For example, it fabricates votes with $i = 4, 5, 7, 8$. CH_2 will find the vote with $i = 4$ is a false vote. So it will set the corresponding bit in Bin_v to 1, let the corresponding bit in Bin_t remain 0 and send it to CH_3. CH_4 will drop the report after it finds another false vote with $i = 7$, as $T_f = 2$ has been reached.

Consider a node in cluster C_1 with $(10, key_{10})$ colludes with CH_0 and gives a vote using key_{10}. This will not raise the chance for CH_0's fabricated report to escape filtering. As an intermediate node finds $\lfloor 10/9 \rfloor = 1 \neq 0$, it will treat it as a false vote.

If a node that owns key_4 has been compromised and launches the false votes on real reports attack. It may give a false vote to a real report of CH_0. If CH_0 still selects votes with $i = 1, 4, 5, 7, 8$, CH_2 will find the vote with $i = 4$ is a false vote. But it won't drop the report as $T_f = 2$ has not been reached yet. The real report could still reach the sink S.

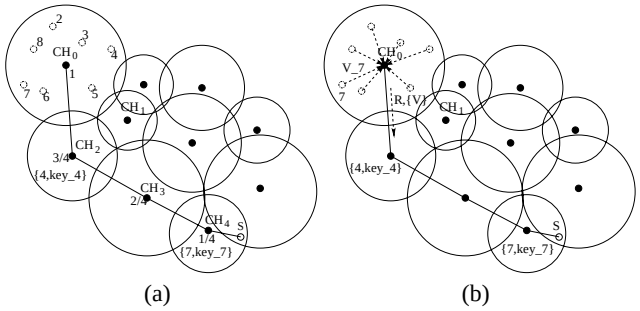

Figure 1: Illustration of PVFS: (a) CH_0's verification key assignment, (b) Report generation and verification

4. ANALYSIS

In this section, we analyze the merits and the performance of PVFS. The analysis results quantify the resiliency, efficiency, and scalability of PVFS. To simplify the analysis, we consider a 2-D terrain, over which N sensor nodes are randomly spread. The sink is located at the center of the terrain.

4.1 General Analysis of PVFS

The cluster-based organization constrains the degree to which compromised nodes could abuse their keys, as it becomes meaningless for nodes in different clusters to collude. To successfully forge a fabricated report that could not be detected, the attacker must collect enough keys, $s - (T_f - 1)$ distinct keys, from one single cluster, because each report must be endorsed by multiple distinct votes using keys bound to one cluster. It also reduces the storage consumption of the keys, as nodes only store their generation key except for the CHs.

The probabilistic key assignment method has three desirable features: First, the verification CHs are chosen in a probabilistic manner. It would become harder for the attackers to avoid being detected as they could not predict which nodes actually are the verification nodes. Second, it would reduce the key storage overhead, because the nodes closer to the sink will have less chance to have a key for remote clusters while they tend to be intermediate nodes of more paths. Third, this method brings higher probability to filter out or accept a report with votes in the first few steps.

We use the voting method to deal with both kinds of attacks. Each intermediate node will not decide whether to drop a report independently. This brings advantages such as when the number of verified true votes reaches the T_t, we would stop further verification and save more energy. But when some intermediate CHs have been compromised, it would also bring more security challenges. When $T_f > 1$, an obvious drop in filtering power occurs. We consider this to be the obligatory cost as we want to deal with these two attacks simultaneously.

4.2 Filtering Effectiveness and Key Storage Overhead

We use the number of hops a forged report can traverse before the first false vote is detected as the first metric. We name it detecting position h_1, which is also the filtering position when $T_f = 1$. Consider a setting where there is a single compromised node (or equivalently, non-colluding compromised nodes). Let node Z be the compromised node, with a distance (in hop count) of d_0 to the sink. A forged report injected by node Z is forwarded along a multi-hop path to the sink, denoted by $Z \rightarrow \ldots \rightarrow CH_i \rightarrow \ldots \rightarrow S$, in

which the $CH_i (1 \leq i \leq d_0)$ are intermediate CHs. Because a compromised node has at most one key for any cluster, it has to forge $s-1$ votes.

THEOREM 1. *The detecting position h_1, defined as the expected number of hops a forged report can traverse before the first false vote is detected is:*

$$h_1 = 1 + \sum_{i=2}^{d_0} (i \cdot p_i \cdot \prod_{j=1}^{i-1} (1 - p_j))$$

where $p_x = \dfrac{(s-1) \cdot (d_0 - x)}{d_0 \cdot L}$

PROOF. The distance from Z to the sink is d_0, and the distance from CH_i to the sink is d_i. We know that $d_i = d_0 - i$. For any intermediate node CH_i, it has a probability of d_i/d_0 to have a key of the cluster detecting the event. The forged report has a probability of $(s-1)/L$ to carry a vote that it claims to be generated by the same key. Therefore, the probability that node CH_i detects on false vote is:

$$p_i = \frac{(s-1)}{L} \cdot \frac{d_i}{d_0}$$

Because all the intermediate CHs perform the same checking task, the probability that a fabricated report elude the check by CH_1 to CH_{i-1} but one false vote be detected by CH_i is:

$$p_i \cdot \prod_{j=1}^{i-1} (1 - p_j)$$

So h_1 could be expressed as above. □

Then we consider there are N_c nodes in a cluster that have been compromised, which is the worst case. If N_c nodes come from different clusters, it is useless for them to collude. If the N_c nodes are located along the path to sink, each of them will only have a probability d_i/d_0 to have a key of the cluster detecting the event.

THEOREM 2. *In the worst-case, with N_c compromised nodes, h_1 is:*

$$h_1 = 1 + \sum_{i=2}^{d_0} (i \cdot p_i \cdot \prod_{j=1}^{i-1} (1 - p_j))$$

where $p_x = \dfrac{(s - N_c) \cdot (d_0 - x)}{d_0 \cdot L}$

Here $N_c < s$, otherwise the detecting ratio would be 0. As we use the voting scheme, our filtering power is quite different from existing en-route filtering methods. It highly depends on the value of T_f, and we may illustrate this point by h_2, which means the expected filtering hops when $T_f = 2$.

$$h_2 = 1 + \sum_{i=2}^{d_0} (i \cdot p_i \cdot \sum_{j=2}^{d_0-i-1} (p_j \cdot \prod_{k \neq j, k=1}^{d_0-i-1} (1 - p_l)))$$

where $p_x = \dfrac{(s - N_c) \cdot (d_0 - x)}{d_0 \cdot L}$

In PVFS, each node stores only one generation key, except for the CHs. We assume that CHs are uniformly distributed in a 2-D terrain to facilitate this analysis. Then we could get the following theorem:

THEOREM 3. *The average number of verification keys stored by an intermediate node is:*

$$N_{v\_key} = c \cdot (d_{max} - d_i)$$

PROOF. In this equation, d_{max} represents the distance between the furthest cluster and the sink. The number of CHs that are h hops away from the sink is $4 \cdot 2h$. Consider how many CHs that are $h + j$ hops away selects the CHs that are h hops away as the intermediate nodes, on the average, the number should be: $\frac{8 \cdot (d_i + j)}{8 \cdot d_i}$ if each CH selects one path. Any CH that is $h + j$ hops away selects a CH that is h hops away and already on the path to the sink with a probability: $\frac{d_i}{d_i + j}$. As each CH selects c paths, the average number of verification keys a CH that is h hops away from the sink stores for the CHs that are $h + j$ hops away from the sink is c. And the total number of verification keys is:

$$N_{v\_key} = \sum_{j=1}^{d_{max} - d_i} c$$

□

Though PVFS does not require uniform distribution of CHs, this theorem still reveals PVFS's advantage on key storage overhead.

4.3 Simulation Evaluation

We consider three existing approaches for comparison: (1) SEF [1], (2) LBRS [3] and (3) our scheme PVFS. All approaches are simulated on a custom simulator which generates random deployment. Nodes are homogeneous and can be deployed in this area arbitrarily. $N = 4000$, the number of clusters $cn = 400$, $L = 10$, $s = 5$, and $T_t = 5$. We consider the following tunable parameters: (1) T_f, the threshold of false votes to drop a report. In our experiments, we vary T_f between 1 and 3. (2) N_c, the number of compromised nodes. (3) d_0, the hops between the report generation cluster to the sink.

As shown in Fig.2 (a), our PVFS is resilient to increasing the number of compromised nodes. Even when $N_c = 10$, two or more compromised nodes belonging to the same cluster rarely occur. So these three lines are very close to each other. It indicates that cluster-organization and probabilistic key assignment effectively constrain the level that the key of a compromised node could be abused.

Fig.2 (b) shows the fact that the filtering power of PVFS drops sharply with the increase of T_f, which is in accordance with our anticipation. When $T_f = 2$, it requires more than twice the number of hops to drop a fabricated report. And the longer a report travels, the lower the probability that an intermediate CH has a verification key for that cluster.

Fig.2 (c) compares the percentage of dropped fabricated reports of SEF [1], and our PVFS ($T_f = 1$), when $N_c = 1$. Our method always has a higher filtering power when the report travels the same hops and $T_f = 1$. When the application only cares about the fabricated report with false votes attack, we would set $T_f = 1$. Our PVFS is a highly efficient filtering scheme in this situation. So we could adjust T_f to accommodate different user scenarios.

Fig.2 (d) illustrates the key storage overhead of SEF [1], LBRS [3], and PVFS. Here we do not use uniform distribution, so the simulation result is different from Theorem 3. But it clearly shows our advantage on both maximum and average key storage overhead. Although more paths would contain CHs closely deployed around the sink, they have a smaller probability to be chosen as verification nodes.

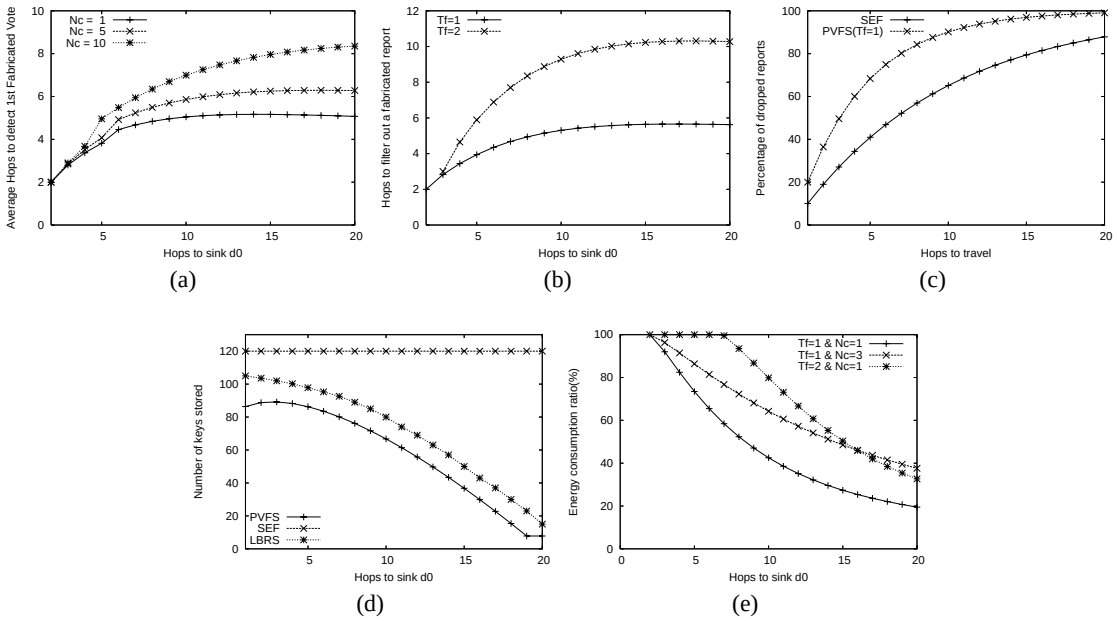

Figure 2: (a) Filtering performance with N_c. (b) H_1 and H_2 when $N_c = 3$. (c) Portion of dropped fabricated reports. (d) Key storage overhead. (e) Energy saving of PVFS.

The early dropping of fabricated reports leads to significant savings of energy in WSNs. Assume that report transmission consumes the same amount of energy between any pair of nodes. Then we can use h'/h to represent the energy consumption ratio between PVFS-protected and unprotected paths. Fig.2 (e) shows that PVFS saves a high percentage of energy that may be consumed by a fabricated report. It also shows that the energy savings of PVFS depends on the location of the generation cluster. Due to our probabilistic key assignment, a fabricated report tends to be purged in the first few steps. When the report generation cluster is far away from the sink, the energy consumption of unprotected WSNs goes up quickly, but the energy consumption of PVFS-protected WSNs remains almost the same.

5. CONCLUSION AND FUTURE WORK

In this paper, we study the fabricated report with false votes attack and the false votes on real reports attack in WSNs. As most of the existing works deal with the first attack while leaving an easy way to launch the second attack, we propose a probabilistic voting-based filtering scheme. Under the framework of the en-route filtering scheme, our PVFS combines cluster-based organization, probabilistic key assignment, and voting methods. Through analysis and simulation, we demonstrate that PVFS could address both attacks while maintaining a sufficiently high filtering power.

We also consider the following extensions as possible future work. First, introduce k-clusters into cluster-based organization, to further improve the resiliency of our scheme. Second, use back check key and drop report acknowledgement together, and modify Algorithm 2 such as to conduct spot-check even after the accepted flag $Flag_r = 1$, to offer stronger protection for compromised intermediate CHs. We will further improve the scheme, by designing and choosing the best methods in these aspects, make it more resilient and efficient.

6. REFERENCES

[1] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical En-route Filtering of Injected False Data in Sensor Networks," IN *IEEE Proceedings of INFOCOM 2004*, 2004, pp.839-850.

[2] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "An Interleaved Hop-by-Hop Authentication Scheme for Filtering False Data in Sensor Networks," IN *IEEE Proceedings of Symposium on Security and Privacy 2004*, 2004, pp.259-271.

[3] H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbaugh, "Toward Resilient Security in Wireless Sensor Networks," In *ACM Proceedings of MobiHoc 2005*, 2005, pp.34-45.

[4] M. Jiang, J. Li, and Y. C. Tay, "Cluster Based Routing Protocol (CBRP) Functional Specification Internet Draft," *draft-ietf-manet-cbrp.txt*, 1999.

[5] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge," In *IEEE Proceedings of INFOCOM 2004*, 2004, pp.597-608.

[6] C. Haowen and A. Perrig, "PIKE: peer intermediaries for key establishment in sensor networks," IN *IEEE Proceedings of INFOCOM 2005*, 2005, pp.524-535.

[7] J. Douceur, "The Sybil Attack," In *Proceedings of International Workshop on Peer-to-Peer Systems (IPTPS) 2005*, 2002.

[8] J. Newsome, E. Shi, D. Song, and A. Perrig, "The Sybil attack in sensor networks: analysis & defenses," In *Proceedings of Third International Symposium on Information Processing in Sensor Networks (IPSN)*, 2004, pp.259-268.

[9] A. Wood and J. Stankovic, "Denial of Service in Sensor Networks," In *IEEE Computer*, vol.35, 2002, pp.54-62.

[10] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," In *ACM Proceedings of MobiCom 2000*, 2000, pp. 255-265.