PVFS: A Probabilistic Voting-based Filtering Scheme in Wireless Sensor Networks

Feng Li*, Avinash Srinivasan and Jie Wu

Department of Computer Science and Engineering,
Florida Atlantic University,
Boca Raton, FL 33431, USA
E-mail: fli4@fau.edu E-mail: asriniva@fau.edu
E-mail: jie@cse.fau.edu
*Corresponding author

Abstract: In this paper, we study two important attacks in Wireless Sensor Networks (WSNs):

- the fabricated report with false votes attack
- the false votes on real reports attack.

Most of the existing works address the first attack while leaving an easy way for the attackers to launch the second. We draw our motivation from this and propose a Probabilistic Voting-based Filtering Scheme (PVFS) to deal with both of these attacks simultaneously. With the general en-route filtering scheme as the underlying model, PVFS combines cluster-based organisation, probabilistic key assignment, and voting methods to curtail these attacks. Through both analysis and simulation, we demonstrate that PVFS can achieve strong protection against both the aforementioned attacks while maintaining a sufficiently high filtering power.

Keywords: cluster-based organisation; en-route filtering; fabricated report attack; false votes attack; probabilistic key assignment; voting method; Wireless Sensor Networks; WSNs.

Reference to this paper should be made as follows: Li, F., Srinivasan, A. and Wu. J. (xxxx) 'PVFS: A Probabilistic Voting-based Filtering Scheme in Wireless Sensor Networks', *Int. J. Security and Networks*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Feng Li received the BS Degree (2002) and MS Degree (2005) in Computer Science from Southeast University, China. He is currently working toward the PhD Degree in the Department of Computer Science and Engineering, Florida Atlantic University, under the supervision of Dr. Jie Wu. His primary research interests include security and trust in WSNs and mobile ad hoc networks. He is a student member of IEEE.

Avinash Srinivasan received the BE Degree with Honors (1999) in Industrial and Production Engineering from University of Mysore, India, and MS Degree (2003) in Computer Science from Pace University, New York, USA. He is currently working toward the PhD Degree in the Department of Computer Science and Engineering at Florida Atlantic University, under the supervision of Dr. Jie Wu. His research interests are security in WSNs and MANETs.

Jie Wu is a Distinguished Professor in the Department of Computer Science and Engineering at Florida Atlantic University and a Program Director at the US National Science Foundation. He has published more than 300 papers in various journals and conference proceedings. His research interests include mobile computing, routing protocols, fault-tolerant computing, and interconnection networks. He is a recipient of the 1996–1997, 2001–2002, and 2006–2007 Researcher of the Year Award at Florida Atlantic University. He served as an IEEE Computer Society distinguished visitor and is the Chairman of IEEE Technical Committee on Distributed Processing (TCDP). He is a senior member of the IEEE.

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, forest fire monitoring, etc. 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, then the 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 drain the finite amount of energy in the battery-powered sensors that forward such fake reports. Detecting and purging fabricated reports injected by compromised nodes is a important research challenge, since the adversary knows all of the security information held by the compromised nodes.

Several recent research efforts, such as SEF (Ye et al., 2004), IHA (Zhu et al., 2004), and LBRS (Yang et al., 2005) have proposed mechanisms to filter out injected fabricated reports in the forwarding process. Their basic idea is as follows: 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. Also, once an incorrect MAC is detected, that report is dropped.

These mechanisms offer the base to solve the fabricated report with false votes attack and keep improving the efficiency. However, 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 SEF, IHA, or LBRS are used, all these true event reports will be dropped during the routing process. Hence, the false votes on real report attack will become a major method to cause denial-of-service in WSNs.

In this paper, we design a scheme that can 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 organisation 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:

- We use a cluster-based organisation in our design. By doing so, compromised nodes from different clusters can not collude. It limits the extent to which the compromised nodes can abuse their keys.
- We propose a probabilistic key assignment. On average, the cluster-head stores less verification keys than the verification nodes in other methods. The false votes tend to be detected in the first few steps.
- We present a voting method. The decision on whether to drop a report depends on whether the recorded result reaches a application specific 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 analyse our scheme and present the simulation results in Section 4. Section 5 introduces two possible extensions to PVFS. Finally, Section 6 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) (Ye et al., 2004), an Interleaved per Hop Authentication scheme (IHA) (Zhu et al., 2004), and a Location-Based Resilient Security solution (LBRS) (Yang et al., 2005), provide some protection against the fabricated report attack. SEF , IHA and LBRS use general en-route filtering framework as the base of their scheme, which is also the underlying model for the PVFS presented in this paper.

2.1 Related work

SEF (Ye et al., 2004) 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. To address this problem in this paper, we use a cluster-based organisation where we divide the network into clusters and bind the generation key to the cluster's ID.

IHA (Zhu et al., 2004) 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 the path are compromised. Second, it relies on deterministic key sharing. Each node must know one predetermined upstream and one predetermined downstream neighbour and establish symmetric keys with them. However, as a deterministic method, IHA cannot be used in a dynamic environment, which restricts its usage for WSN as sensors could sleep or die. IHA presents us with the idea of associating nodes along a routing path together and also leads us to using probabilistic key sharing in our design.

In LBRS (Yang et al., 2005), the sensing field 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. Although LBRS improves the resilience of en-route filtering, it requires additional information such as sensor location information.

There are obvious innovations in these three mechanisms, though, all of them make false votes injection attack easier while offering protection against the fabricated report attack. This paper aims to find a way of addressing 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.

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, in terms of the detection percentage for forged reports, 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 the detecting and dropping of all forged reports. The sink serves as the final guard in rejecting any forged reports that get through the intermediate nodes.

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 organisation, probabilistic key assignment, and voting method. Notations used in this paper are listed in Table 1.

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 index i
s	Required number of votes for a legitimate report
d_0	The distance from the report generation cluster head to sink
d_i	The distance from ith verification node to sink
T_t	Threshold of verified true votes to accept a report
T_f	Threshold of verified false votes to drop a report

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 can not generate enough true votes for a fabricated report. If the adversary manages to capture majority of the sensors, there is no need for the adversary to launch the above two attacks and consequently there will be no solution for filtering.

All existing routing protocols in WSNs may be grouped into one of the following three categories: one-hop model, multi-hop planar model, and cluster-based hierarchical model. In the one-hop model, every node directly communicates with the base station. It is considered to be impractical for large scale WSNs, and filtering methods are not applicable to this model. In the multi-hop planar model, a report travels from source to destination hop by hop until it arrives at the destination. SEF (Ye et al., 2004) and LBRS (Yang et al., 2005) construct their filtering methods on this model. As there is no division of sensors in this model, attackers in arbitrary locations may cooperate with each other to fabricate reports in SEF (Ye et al., 2004), or we need additional information such as the location information in LBRS (Yang et al., 2005) to divide the WSN and differentiate nodes.

The cluster-based model is naturally suitable for the filtering mechanisms. It breaks the network into clusters. We choose the cluster-based model to organise 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 denoted as CH. Each cluster has a unique cluster ID C_{id}. There are many ways to form clusters, elect CHs, and generate unique cluster IDs (Jiang et al., 1999). We use a simple cluster formation method in our scheme: a node with the smallest node ID in its one-hop neighbourhood is elected as the CH, and its one-hop neighbours join that cluster. The node ID is unique and predetermined before sensor deployment.

We assume all the sensors within the same 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 the regular nodes and discovers a route to the sink which only consists of CHs. Each CH discovers c paths to the sink to overcome node failure.

As already mentioned, 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 (Marti et al., 2000) to solve these attacks. A detailed discussion on all possible attacks and the counteractions are beyond the scope of this paper. To simplify the discussion, we also assume that after sensor deployment, for a very shot period of time, no node is compromised. The cluster formulation, key distribution, and route discovery is completed during this time period without being attacked.

3.2 Initialisation and key assignment

PVFS requires a pre-generated global key pool of n keys $\mathcal{K} = \{K_i : 0 \le i \le n-1\}$. After the sensors have been organised into clusters, we divide the global key pool into cn non-overlapping partitions $\mathcal{K}_{C_{id}} = \{K_i : L \cdot C_{id} \le i \le L \cdot (C_{id}+1) - 1\}$. Here L represents the number of keys in each partition and cn is the number of clusters in the WSN, $L \ge s$ 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. The partition to which a particular key K_i belongs to is determined 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 Du et al. (2004a) 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 are 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 given as follows:

$$P = \frac{d_i}{d_0}.$$

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 initialisation 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 Du et al. (2004a) 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 nearby clusters compete with each other according to application specific rules, 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 utilise methods such as radio resource testing (Newsome et al., 2004) to ensure that each node could cast only one vote for each report to protect our system from the Sybil attack (Douceur, 2002), 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 neighbours. Algorithm 1 demonstrates the 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 each 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 to confirm that it holds the corresponding verification key and records the verification result using two binary sequences Bin_v and Bin_t. Bin_v and Bin_t 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 that is being forwarded will be of 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 stored in a CH; $D_{K_i}(Vote)$ denotes to decrypt a vote with K_i; $E_{K_i}(R)$ means to encrypt a report with K_i.

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

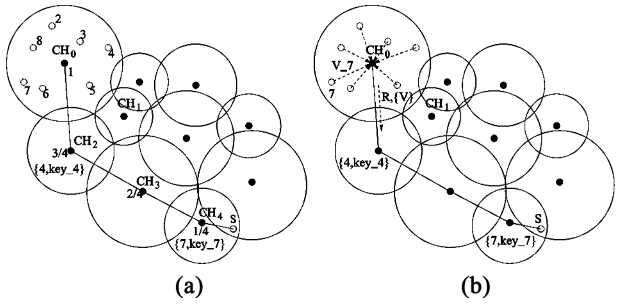
Algorithm 2 ReportVeri($R, DS_K, T_f, T_t, K_{bc}$)

```
1:  if $R.Flag_r = 1$ then
2:      Forward $R$, EXIT;
3:  end if
4:  for each vote $(i, K_i)$ in $R.\{Vote\}$ and $R.Bin_v = 0$ do
5:      if $\lfloor i/L \rfloor \neq C_{id}$ then
6:          $R.Bin_v[i] \leftarrow 1$;
7:      else if $(i, K_i) \in DS_K$ then
8:          if $D_{k_i}(Vote) = H(R.Report)$ then
9:              $R.Bin_t[i] \leftarrow 1$;
10:         end if
11:         $R.Bin_v[i] \leftarrow 1$;
12:     end if
13: end for
14: Count the number of verified true and false votes;
15: Drop $R$ and EXIT if $T_f$ has been reached;
16: $R.Flag_r \leftarrow 1$ if $T_t$ has been reached;
17: Add $E_{K_{bc}}(H(R))$ to $R.\{SIG\}$;
18: Forward $R$, EXIT
```

3.5 Example

Figure 1 illustrates our idea clearly. All nodes report to the sink S. We consider $L = 9$, $s = 5$ and $T_f = 2$. In the initialisation phase, all sensors are organised into clusters. In Figure 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. Figure 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 $(7, key_7)$.

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



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 sends 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 forwards it.

Let us consider another situation. If CH_0 fabricates a report about a non-existing event, it has to fabricate some votes. For illustration, assume 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, but lets the corresponding bit in Bin_t remain 0 and send it to CH_3. CH_4 finds another false vote with $i = 7$, and drops the report 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 improve the chance for CH_0's fabricated report to elude 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, then 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 that 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 can still reach the sink S.

4 Analysis

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

4.1 General analysis of PVFS

The cluster-based organisation 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 distinct keys) from one single cluster, because each report must be endorsed by multiple distinct votes using keys bound to a single cluster. It greatly reduces the storage overhead of the keys, as nodes will only store their generation key except for the CHs. It also makes the identification and exclusion of compromised nodes easier. After we detect a false vote, we can find out which node it claims to be generated from and which cluster the node belongs to, and then isolate that node.

Some concerns may rise as the cluster-based organisation makes the WSN sparser, the path from the location where the event happens to the sink may become shorter, and filtering ratio may drop. Remember the primary goal of the filtering methods is to discard the fabricated report as quickly as possible. The longer it travels, the more resources it wastes. So, although the path may become shorter after we use the cluster-based organisation (actually in most cases it will not), it will not affect our primary goal.

Our probabilistic key assignment method has three desirable features: First, the verification CHs are chosen in a probabilistic manner, so it is harder for the attackers to avoid being detected as they can not predict which nodes are actually the verification nodes. Second, it will reduce the key storage overhead, because the node closer to the sink will have less chance of having a key for the remote cluster while they tend to be an intermediate node for more paths. We will further prove this in Section 4.2. Third, this method brings higher probability of filtering out or accept a report with votes in the first few hops.

We use the voting method to deal with both kinds of attacks. Now each intermediate node will not decide whether to drop a report independently. This may bring some advantages such as when the number of verified true votes reaches 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 inevitable cost as we want to deal with these two attacks simultaneously.

4.2 Filtering effectiveness and key storage overhead

For the scenario with a single compromised node, we use the number of hops a forged report can traverse before the first false vote is detected as the first metric. We call it the detecting position h_1, which is also the filtering position when $T_f = 1$.

Consider a base 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 $Z \to CH_1 \to \ldots \to CH_i \to \ldots \to Sink$, in which the d_0 nodes $CH_i (1 \le i \le d_0)$ are intermediate CHs. Because a compromised node has at most one key for any cluster, it has to forge at least $s - 1$ votes.

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

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

where $p_x = \frac{(s-1)\cdot(d_0 - x)}{d_0 \cdot L}$ and x denotes i or j.

The distance from Z to sink is denoted as d_0, and the distance from CH_i to sink is denoted as d_i. We know that $d_i = d_0 - i$.

For any intermediate CH_i, it has a probability of d_i/d_0 to have a symmetric verification key of the cluster detecting the event, thus it may be able to verify one vote for that report. Therefore, the probability that node CH_i detects a 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 be detected by CH_i for one false vote is:

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

So the expected number of hops is as follows:

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

where $p_x = \frac{(s-1)\cdot(d_0 - x)}{d_0 \cdot L}$ and x denotes i or j.

Then we consider there are N_c nodes in a cluster that have been compromised, which is the worst case we may have. Because, if N_c nodes come from different clusters, it is useless for them to collude as we motioned before. If these N_c nodes are located along the path to sink, each of them will have a probability d_i/d_0 to have a key of the cluster detecting the event, which is still not the worst case.

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

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

where $p_x = \frac{(s-N_c)\cdot(d_0 - x)}{d_0 \cdot L}$ and x denotes i or j.

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 is the expected filtering hops when $T_f = 2$.

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

where $p_x = \frac{(s-N_c)\cdot(d_0-x)}{d_0 \cdot L}$ and x denotes i or l.

In PVFS, each node stores only one generation key, except for the CHs. CHs also have the probability $P = d_i/d_0$ to store one generation key for each remote cluster that use them as intermediate nodes to the sink. We assume that CHs are uniformly distributed in a 2-D terrain to facilitate this analysis. Then we can get the following theorem:

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

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

In this equation, $d_{\max}$ represent the distance between the sink and the furthest cluster. The number of CHs that are h hops away from the sink is $4 \cdot 2h$. Assume each CH selects one path to send reports to the sink. The average number of CHs that are $h + j$ hops away from the sink select one CH that is h hops away from the sink, to be the intermediate nodes on the path is: $\frac{8\cdot(d_i+j)}{8\cdot d_i}$. As we design in the probabilistic key assignment, any CH that are $h + j$ 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 is $h + j$ hops away from the sink is 1. And the total number of verification keys is:

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

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

4.3 Simulation evaluation

We conduct a simulation study to evaluate the performance of the proposed PVFS scheme. We consider three approaches for comparison:

- SEF (Ye et al., 2004),
- LBRS (Yang et al., 2005)
- PVFS.

All approaches are simulated on a custom discrete event simulator, which generates random initial deployment. We set up the simulation in a square area as our target field. We assume nodes are homogeneous and can be deployed in this area arbitrarily. The total number of sensors N is 4000, the number of cluster-heads is 400, number of keys bound to a cluster L is 10, the required vote for a report $s = T_t = 5$. In the simulation, we consider the following tunable parameters:

- T_f, the threshold of false votes to drop a report. In our experiments we vary T_f between 1 and 3.
- N_c, the number of compromised nodes.
- d_0, the hops between the report generation cluster to the sink.

As shown in Figure 2(a), our PVFS is resilient to the increasing number of compromised nodes. That clearly shows the effect of cluster-based organisation and probabilistic key assignment. Although when $N_c = 10$ there are a large number of compromised nodes, two or more compromised nodes rarely belong to the same cluster. So these three curves are very close to each other. It indicates that cluster-organisation and probabilistic key assignment effectively constrain the extent to which the key of a compromised node is abused.

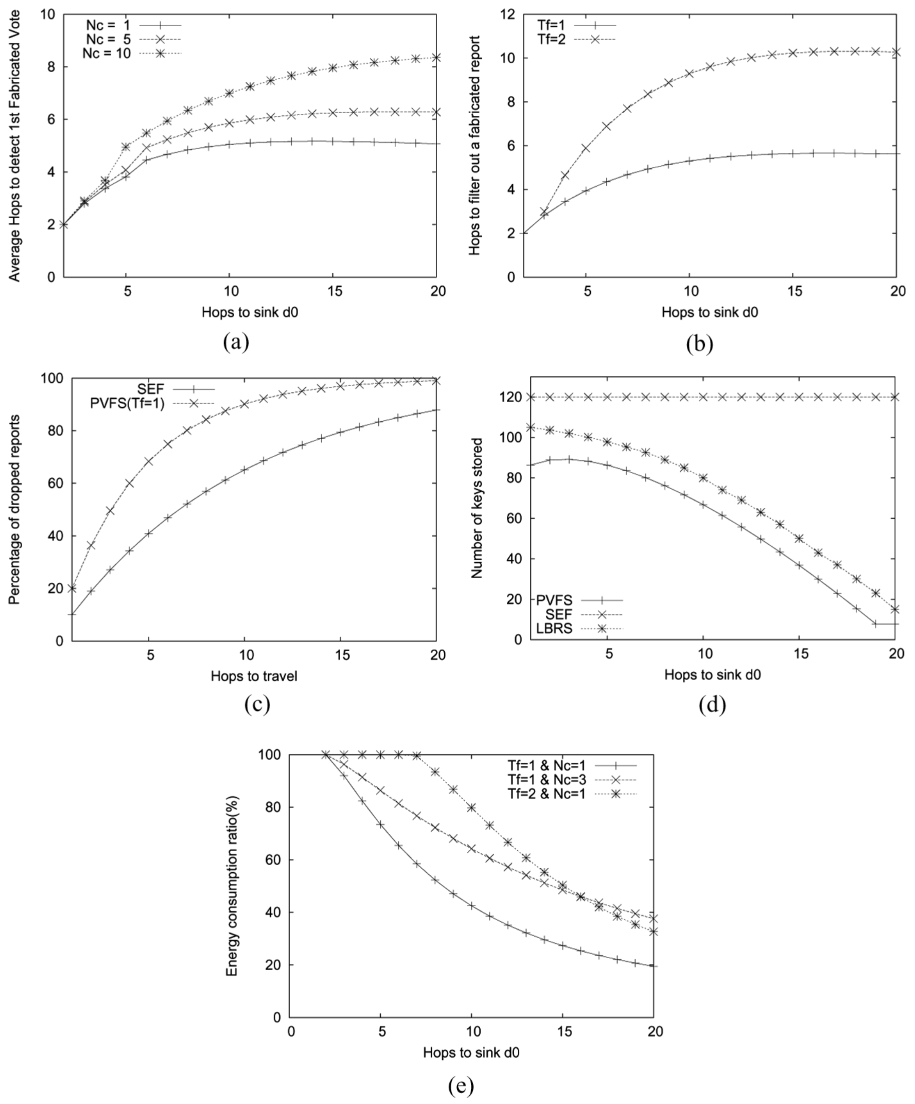
Figure 2(b) shows that the filtering power of PVFS drops sharply with the increasing T_f, which is in accordance with our anticipation. When $T_f = 2$, it requires more than twice the number of hops to drop a fabricated report. Because when $T_f = 2$, after a verification node detects a false vote, it will not drop the report but keep forwarding it until another false vote has been detected. And the longer a report travels, the lower the probability that a intermediate CH has a verification key for that cluster.

Figure 2(c) compares the percentage of dropped fabricated reports of SEF (Ye et al., 2004), and our PVFS ($T_f = 1$), when $N_c = 1$. Our method always has a higher filtering power when the report travels the same number of hops and $T_f = 1$. When the focus is only on the fabricated report with false votes attack, we would set $T_f = 1$. Our PVFS is also a highly efficient filtering scheme for this situation. So we could adjust T_f to accommodate different user scenarios.

Figure 2(d) illustrates the key storage overhead of SEF (Ye et al., 2004), LBRS (Yang et al., 2005), and our PVFS. Here we do not use the uniform distribution of CHs, so the simulation result is different from Theorem 3. But it still clearly shows our advantage over both maximum and average key storage overhead. As we can see from the figure, the maximum number of verification keys is stored by those CHs at the intermediate distance to the sink. Although more paths would contain those CHs closely deployed around the sink, they have a smaller probability to be chosen as verification nodes.

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. Figure 2(e) compares the percentage of dropped fabricated reports of

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 and (e) energy saving of PVFS



SEF (Ye et al., 2004), 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 $T_f = 1$. Figure 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 report 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 Extension

The cluster-based organisation, probabilistic key assignment, and voting are three main components of PVFS. To keep the logic clear, we have only discussed these three main components in the previous sections. In this section, we will explain some extensions which would make PVFS more adaptive to dynamic environments and more secure. One extension is to accommodate reports with different priority using dynamic thresholds. The other is to exploit time-constraint keys.

5.1 *Priority-oriented thresholds*

In sensor networks, different types of reports have different value. For example, in a forest fire monitoring system, a hourly report that state the average temperature in that hour is 70 F, and a emergency report that state the detected temperature is 350 F clearly have different significance. For reports with different priorities, assigning the same s and same T_t/T_f is both dangerous and unfair. Assuming all nodes have the same criteria to decide reports' priority, we should design a priority adaptive threshold for voting and filtering. On the other hand, different cluster can have different number of sensors, i.e., different L, given the factor that sensors are not always uniformly deployed. This also requires that we use the adaptive thresholds.

The adaptive thresholds method requires some changes in the implementation details of the PVFS method. First, when the intermediate cluster-heads get verification keys from nodes in the original cluster, they should also get information about the number of nodes in the cluster. Second, the WSN should specify a unique policy to map priority with s. Instead of providing the required number of votes for a report, the policy should specify the required percentage of votes to the number of nodes in a cluster, i.e., $s = \lfloor f(Priority) \cdot L \rfloor$, where $f(Priority) \in [0, 1]$ and is non-decreasing with $Priority$. Therefore, reports that have higher priority will require more votes. Third, T_t/T_f should also be specified as a percentage of s. There are different choices to determine T_t/T_f. When they are priority-oriented, where $T_t = \lfloor g_1(Priority) \cdot s \rfloor$ and $T_f = \lfloor g_2(Priority) \cdot s \rfloor$, the mapping function g_1 and g_2 should be defined according to the application goal.

5.2 *Time-constraint keys*

The second extension is to use time-constraint keys. In SEF, IHA, and LBRS, the symmetric keys are securely distributed only once immediately after deployment and they never expire. Even when the sink detects that a intermediate node has been compromised by our adversary, we can only revoke the generation key of that compromised node but not the verification keys. Therefore, our adversary can accumulate verification keys and finally break the protection.

One method to counter this problem is using the the time-constraint keys. We use TESLA (Perrig et al., 2000, 2001) key chain to realise the time-constraint. When each node gets its generation key in the key assignment phase, it will generate a key chain by using the generation key as the root key: $K_i^0 = K_i, K_i^1 = H(K_i^0), K_i^2 = H(K_i^1) \ldots K_i^n = H(K_i^{n-1})$. When an intermediate node has been selected as the verification node, the node with the generation key will send the last key in the key chain K_i^n to the intermediate node as the verification key. The policy of the WSN specifies n which decides the length of the key chain and the length T of the valid time interval of the verification keys.

At the end of a valid time interval, the node with the generation key will send the new verification key K_i^{n-1} to the sink. The sink should check whether some of the verification nodes have been detected as compromised. For those verification nodes that have not been detected as compromised, the sink will send the updated verification key $E_{K_j}(K_i^{n-1})$. Therefore, the verification keys on the detected compromised nodes will expire. This method can further restrict compromised nodes, but it also incurs a higher overhead.

6 Conclusion and future work

In this paper, we study the fabricated reports 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 PVFS to efficiently address both these attacks. Under the framework of en-route filtering scheme, our PVFS combines cluster-based organisation, probabilistic key assignment, and voting method together. Through both analysis and simulation, we demonstrate that our PVFS can deal with false votes on real reports attack while maintaining certain filtering power.

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

Acknowledgment

This work was supported in part by NSF grants CCR 0329741, CNS 0422762, CNS 0434533, CNS 0531410, and CNS 0626240.

References

Douceur, J. (2002) 'The Sybil attack', *Proc. International Workshop on Peer-to-Peer Systems (IPTPS)*. AUTHOR PLEASE SUPPLY PAGE RANGE.

Du, W., Deng, J., Han, Y., Chen, S. and Varshney, P. (2004a) 'A key management scheme for wireless sensor networks using deployment knowledge', *Proc. IEEE INFOCOM*, pp.597–600. AUTHOR PLEASE CHECK IF THE CITATION FOR THIS REFERENCE IS OK.

Jiang, M., Li, J. and Tay, Y. (1999) 'Cluster based routing protocol (CBRP)', *Functional Specification Internet Draft*, draft-ietf-manet-cbrp.txt.

Marti, S., Giuli, T., Lai, K. and Baker, M. (2000) 'Mitigating routing misbehavior in mobile ad hoc networks', *Proc. MobiCom*, pp.255–265.

Newsome, J., Shi, E., Song, D. and Perrig, A. (2004) 'The Sybil attack in sensor networks: analysis & defenses', *Proc. Third International Symposium on Information Processing in Sensor Networks (IPSN)*, pp.259–268.

Perrig, A., Canetti, R., Song, D. and Tygar, J. (2001) 'Efficient and secure source authentication for multicast', *Proc. Network and Distributed System Security Symposium (NDSS)*. AUTHOR PLEASE SUPPLY PAGE RANGE.

Perrig, A., Canetti, R., Tygar, J. and D. Song (2000) 'Efficient authentication and signing of multicast streams over lossy channels', *Proc. IEEE Symposium on Security and Privacy*. AUTHOR PLEASE SUPPLY PAGE RANGE.

Yang, H., Ye, F., Yuan, Y., Lu, S. and Arbaugh, W. (2005) 'Toward resilient security in wireless sensor networks', *Proc. ACM MobiHoc*, pp.34–45.

Ye, F., Luo, H., Lu, S. and Zhang, L. (2004) 'Statistical en-route filtering of injected false data in sensor networks', *Proc. IEEE INFOCOM*, pp.839–850.

Zhu, S., Setia, S., Jajodia, S. and Ning, P. (2004) 'An interleaved hop-by-hop authentication scheme for filtering false data in sensor networks', *Proc. IEEE Symposium on Security and Privacy*, pp.259–271.

Bibliography

Chan, H., Perrig, A. and Song, D. (2004) 'Random key predistribution schemes for sensor networks', *Proc. IEEE Symposium on Security and Privacy*, pp.197–213.

Du, W., Deng, J., Han, Y. and Varshney, P. (2004) 'A pairwise key pre-distribution scheme for wireless sensor networks', *Proc. ACM CCS*, pp.228–258.

Heinzelman, W., Chandrakasan, A. and Balakrishnan, H. (2004) 'Energy-efficient communication protocol for wireless microsensor networks', *Proc. Hawaii International Conference on System Sciences (HICSS)*, Vol. 2, pp.10–12.

Liu, D. (2007) 'Resilient cluster formation for sensor networks', *Proc. ICDCS*. AUTHOR PLEASE SUPPLY PAGE RANGE.

Pryzdatek, B., Song, D. and Perrig, A. (2003) 'SIA: secure information aggregation in sensor networks', *Proc. ACM SenSyS*, pp.255–265.

Wood, A. and Stankovic, J. (2002) 'Denial of service in sensor networks', *IEEE Computer*, Vol. 35, No. 10, pp.54–62.

Ye, F., Yang, H. and Liu, Z. (2007) 'Catching moles in sensor networks', *Proc. ICDCS*. AUTHOR PLEASE SUPPLY PAGE RANGE.