The Design and Evaluation of Interleaved Authentication for Filtering False Reports in Multipath Routing WSNs

Youtao Zhang [a,*], Jun Yang [b], Hai T Vu [c], Yizhi Wu [d]

[a] *Computer Science Department, University of Pittsburgh, Pittsburgh, PA 15260*

[b] *Electrical and Computer Engineering Department, University of Pittsburgh Pittsburgh, PA 15261*

[c] *Computer Science Department, University of Texas at Dallas, Richardson, TX 75083*

[d] *Institute of Information Science, Donghua University, Shanghai, China*

Abstract

In this paper, we consider filtering false reports in braided multipath routing sensor networks. While multipath routing provides better resilience to various faults in sensor networks, it has two problems regarding the authentication design. One is that, due to the large number of partially overlapped routing paths between the source and sink nodes, the authentication overhead could be very high if these paths are authenticated individually; the other is that false reports may escape the authentication check through the newly identified node association attack. In this paper we propose enhancements to solve both problems such that secure and efficient authentication can be achieved in multipath routing. The proposed scheme is $(t+1)$-resilient, i.e. it is secure with up to t compromised nodes. The upper bound that a false report may be forwarded in the network is $O(t^2)$.

Key words: Authentication, Sensor Networks, False Report Filtering, Security

1 Introduction

The wireless sensor network has emerged as a promising computing model for many applications including those running in hostile environments e.g. tracking enemy targets in a battlefield. Since sensor nodes are usually left unattended after deployment, they are vulnerable to varying forms of security attacks [4]. Once a sensor node is captured, the sensitive information stored in the node is exposed. In addition, the compromised node may be used to launch further attacks.

Recently the false report injection attack has gained the attention [10,13,9]. In such an attack, a compromised node injects false reports or modifies its relayed reports. Without detecting such reports, the sink may reach a suboptimal or even wrong decision. In addition, routing false reports to the sink consumes the limited energy of relay nodes on the routing path and reduces the lifetime of the network. Several schemes [10,13,9] have been proposed to defend this attack. To prevent accepting false reports at the sink, each report usually encloses a MAC (message authentication code) using the private key that is only known to the sink node and the corresponding reporting sensor. In this way, only those reports generated by corresponding sensors can be accepted. To save the routing energy and thus prolong the lifetime of the network, en-route authentication strategies can be employed such that false reports are detected and dropped early in the routing path. The authentication is performed between two sensors that share the same authentication key. The key can be set up by randomly selecting from a key pool [10], or creating pairwise authentication keys in an interleaved approach [13], or combining location information [9], or periodically refreshing the keys [12]. While each relay sensor has limited authentication ability, a number of consecutive sensor nodes can confidently detect and drop false reports in several hops.

In this paper, we consider filtering false reports in a multipath routing based sensor network. Similar as [10,13,9], we focus our discussion on detecting and dropping false reports early along the routing path to save energy. Multipath routing has gained its popularity as it adapts better to various faults such as sensor failure and signal conflicts. In multipath based routing, the source node sends the reports back to the sink along several paths that may and may not be disjoint. Even

* Corresponding author.
Email address: zhangyt@cs.pitt.edu (Youtao Zhang).

some relay nodes on some paths failed, the report can still be routed to the sink as long as there exists one well-behaved path. Studies have showed that several braided (partially disjoint) paths can achieve better tradeoff among factors such as packet delivery, energy consumption and failure node recovery [1].

In a sensor network with braided multipath routing [1], the number of possible paths between the sink and source nodes are typically large. The false report detection is hence complicated as dropping them along one path does not effectively stops the attack. False reports may escape the authentication check either through a different path or through malicious node association. While the interleaved authentication for single path routing [13] can be adopted, the overhead is probably high. To address this problem, we make the following contributions in this paper:

- We identify a new type of attack – the node association attack in both single path routing and multipath routing scenarios. This attack is more severe in a multipath routing environment due to dynamic node re-association. We elaborate and explain the attack using examples. We then propose to defend the attack with sufficient information included in the ACK message at the node association stage.
- We propose a $(t+1)$-resilient interleaved authentication method for multipath routing. Nodes are associated conservatively in multipath routing. We maintain similar en-route authentication overhead and show that the false report can be forwarded at most $O(t^2)$ hops where t is the number of nodes in the shortest path.

The remainder of the paper is organized as follows. Section 2 reviews the related work and in particular the interleaved hop-by-hop authentication [13] and the braided multipath routing [1] that our scheme is based on. Section 3 details the problems that we target to solve. Section 4 discusses our algorithm, its security analyses, and the enhancement to minimize per report authentication overhead. The experimental results are shown in section 5. Section 6 concludes the paper.

2 Background

2.1 False reports filtering

In this section we first briefly describe the state-of-art false report filtering schemes and then describe in more detail the interleaved hop-by-hop authentication (IHA) [13] for single path routing.

Ye *et al.* [10] proposed a statistical en-route filtering (SEF) scheme in which each node randomly picks up a subset of keys from a global key pool. A report is endorsed with multiple such keys and authenticated by relay nodes who have at least one of endorsing keys. The false report is dropped if a mismatch is found. Zhu *et al.* [13] proposed IHA to associate nodes on a routing path interleavingly such that each node can authenticate the MAC generated by its association node. Yang *et al.* proposed a scheme [9] to achieve high resilience in terms of the number of compromised nodes through incorporating location information in generating authentication keys. By periodically updating the keys with the help from neighboring nodes, Zhang *et al.* [12] proposed a scheme to minimize the harm of compromised nodes.

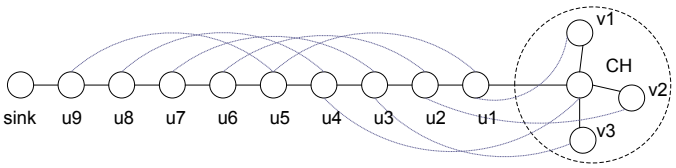


Fig. 1. The IHA scheme (graph adapted from [13], resilient to up to 3 compromised nodes (t=3)).

We now briefly describe the IHA scheme [13] in which nodes are associated and MACs are checked within association pairs. Figure 1 depicts such an association that can achieve $(t+1)$ resilience where $t=3$. Nodes $v1$ to $v3$ are nodes within a cluster with the cluster head CH. $u1-u9$ are relay nodes that forward reports from the CH to the sink. Basically, a node is associated with an upstream (towards the sink) and a downstream (towards the source) node of $t+1$ hops away. For example, $u5$ is associated with $u9$ and $u1$. A unique pairwise key is used in each association, e.g., the $u1-u5$ pair uses $K_{u1.u5}$ and $u5-u9$ pair uses $K_{u5.u9}$. When $u5$ receives a report, it authenticates the MAC generated by $u1$ using $K_{u1.u5}$. Upon success, $u5$ replaces this MAC with a new one using $K_u5.u9$. The new MAC is to be authenticated by $u9$. As we can see, a report needs to carry sliding $t+1$ MACs computed from keys corresponding to $t+1$ association. If any t nodes in this path are compromised, i.e., t keys are exposed, the last association will guarantee that a faulty report be detected because its key is still hidden. To be complete, the nodes with less than $t+1$ hops from the cluster head are associated with distinct nodes from within the cluster, e.g., $u1$ is associated with $v1$. The nodes apart from the sink for less than $t+1$ hops do not need to associate with upstream nodes since there are less than $t+1$ of them and they are unable to conspire to fool the sink. A false report in IHA can travel $O(t^2)$ hops in the network.

	Compromised Nodes			
	t=1	t=2	t=3	t=4
SEF	5.5	10.4	18.5	46.2
IHA	1	4	9	16

Fig. 2. The average number of hops that a false report can travel

We adopt IHA due to its explicit and tight upper bound that a false report can be forwarded in the network. Fig. 2 lists the average number of hops that a false report can travel with different number of compromised nodes. We use the same settings as that in [10] in collecting the data for SEF.

2.2 Multipath routing

The design of different routing techniques in wireless sensor networks (WSNs) is largely influenced by non-traditional factors [5] such as energy consumption, network dynamics, data reporting/aggregation model, fault tolerance etc. For example, directed diffusion routing [3], geographic adaptive fidelity routing [8] forward packets to a subset of nodes towards the sink such that the energy consumption is reduced and the network lifetime is prolonged.

Multipath routing schemes [1,6,11] maintain multiple alternative paths between the sink and the source nodes in order to adapt to link congestion and node failures in the network. Next, we briefly review the *braided multipath routing* which achieves better failure resilience and energy consumption compared to disjoint multipath routing schemes.

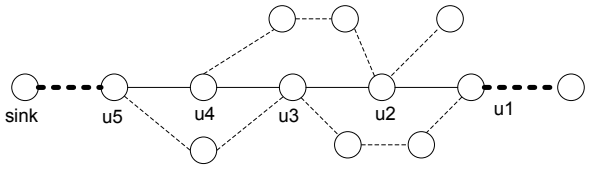


Fig. 3. The braided multipath routing.

Developed from the directed diffusion routing scheme [3], braided multipath routing [1] maintains a primary path and several alternative paths between the sink and the source (cluster head) nodes. In Fig. 3, the primary path is shown by the solid line and the alternative paths are shown by dashed lines. When the sink is to reinforce the routing paths from the sink to the cluster head, each node on the primary path identifies an alternative next node in addition to the one on the primary path; each node on the alternative path only identifies one next node, in the same way as the directed diffusion routing. The alternative paths may join the primary path partially such that the primary and alternative paths are braided with nodes overlapping with each other. As we can see, the number of possible paths between two distant nodes is large. In Fig. 3, for example, there are five possible paths from node $u1$ to node $u5$. Ganesan *et al.* [1] showed that the number is proportional to the n^{th} Fibonacci number if there are n nodes on the primary path. The braided multipath scheme achieves better resilience to node failures simply because the high number of possible paths from the source to the sink. We will explain next our design based on this scheme.

A concern regarding multipath routing is that it may consume more energy to route one packet from the source to the sink. However study [3,1] shows that in a wireless network with probabilistic link or node failures, it is much more energy-efficient to use multipath routing since otherwise a packet needs to send multiple times before it can reach the destination.

3 Problem Statement

In this section we elaborate the two problems that we are to solve in this paper. Before the discussion we define some terms that we use in the paper.

Definition 1 *Two nodes V_1 and V_2 are* **directly associated** *if they are associated with each other for authentication.*

Definition 2 *Two node V_1 and V_2 are* **indirectly associated** *if there is a non-empty list of nodes [X_1,...,X_n] such that V_1 is directly associated with X_1, X_i is directly associated with $X_{i+1}(1 \leq i \leq (n-1))$, and X_n is directly associated with V_2.*

Definition 3 *A node V_1's* **authentication chain***, abbreviated as V_1-chain, consists of all nodes that are directly and indirectly associated with V_1.*

Definition 4 *Assume there are t interleaved authentication chains along the routing path, and two nodes V_1 and V_2 are on the same authentication chain. If between V_1 and V_2, we have $c_i(1 \leq i \leq t)$ nodes respectively from each authentication chain, and $c_j = min(c_1, c_2, ..., c_t)$, then we say V_1 and V_2 are $(c_j + 1)$* **raw rounds** *(abbreviated as rounds) away from each other.*

For example in Fig. 4, nodes $v3$, $u3$ and $u7$ are on the $v3$ authentication chain. nodes $u3$ and $u7$ are directly associated while nodes $u7$ and $v3$ are indirectly associated. Node $u7$ is two rounds away from $v3$.

3.1 Problem 1: malicious node association manipulation

The effectiveness of the interleaved hop-by-hop (IHA) authentication scheme relies on the correctness of the node association, i.e., the $(t+1)$ MACs can be generated and verified alternatively. In the original scheme the node association is established at the beginning stage of each epoch of transferring sensor reports. Fig. 4 depicts the association process. A HELLO message is first sent from the sink to the cluster head *CH* (source node), followed by a reply ACK message from *CH* to the sink. Each message contains a *node list* consisting of up to $(t+1)$ nodes that are used in the node association. These nodes are the $(t+1)$ nearest neighbors in one direction of any node in the path. Hence, the list is a sliding window of the nodes that the HELLO/ACK message passes through.

When the HELLO message is propagated from $u8$ to *CH* in Fig. 4, a node list grows from an empty set in $u8$ to a set of i, $i \leq (t+1)$ nodes that indicates the last i nodes visited in that order. A node N finds its up-stream associated node M by reading the head of the list. The list is then modified by removing M and inserting N at the end, and passed on to the next node. Note that nodes within $(t+1)$ hops from the sink has no up-stream association nodes. When *CH* receives the HELLO message, it replies back with an ACK message by forming a node list containing itself and t sensors in its

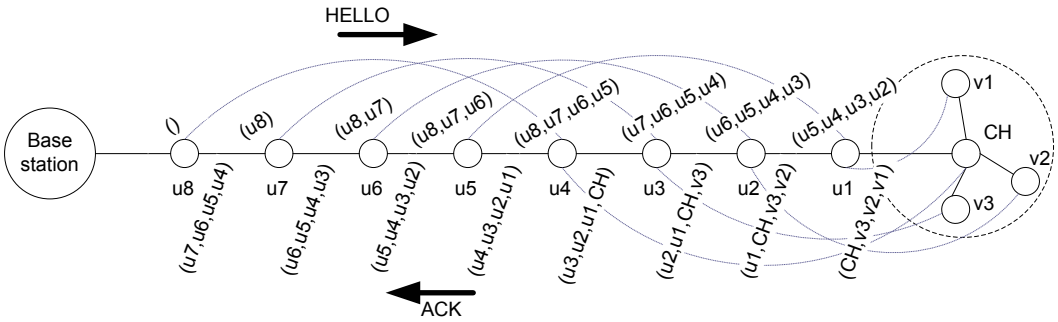


Fig. 4. The node association in IHA scheme.

cluster. Those sensors are to be associated with relay nodes that are within $t+1$ hops from CH. A node N finds its down-stream associated node S by reading from the tail of the list. The list is then updated by removing S and inserting N at the beginning, and passed on to the next node. The nodes in pair associated in this way are $(t+1)$ hops away from each other.

Unfortunately the important node association phase is not rigorously protected from malicious attacks. A compromised node on the route can manipulate the node list and deceive the nodes down in the path. This is serious since, due to the varying nature of sensor routing, nodes in sensor networks may have to be re-associated after each epoch. At this point some sensor nodes may have already been compromised and thus can initiate the attack to perform malicious node association manipulation.

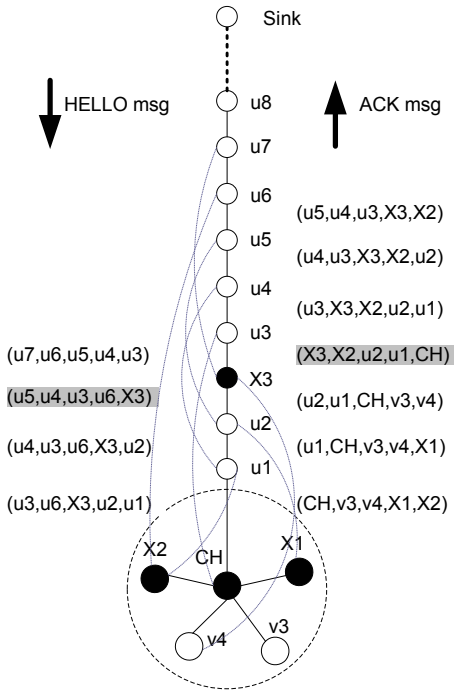


Fig. 5. The node association attack (t=4).

Next we elaborate the attacking procedure using an example in Fig. 5 where t=4. That is, if there are four compromised nodes $X1, X2, X3$, and CH, the scheme should still be secure. The HELLO and ACK messages also include a node list with 5 node IDs. For clarity, we omit the HELLO messages received by $u3-u8$ and the ACK messages beyond $u6$ as they are not relevant in the attack. Ideally the two nodes in each association pair should be 5 (=t+1) hops away from each other.

Let us focus on $X3$ who manipulates the node association. $X3$ first receives a HELLO message with the list $(u7,u6,u5,u4,u3)$. Therefore $X3$ is up-stream associated with $u7$. However, since it is compromised, it sends $u2$ a fake node list $(u5,u4,u3,u6,X3)$. This list ensures that $u2$ and $u1$ are up-stream associated with $u5$ and $u4$ respectively while the correct association should be $u6$ and $u5$. Similarly, in processing the ACK message, $X3$ forges the outgoing node list to $u3$. The fake node list ensures that $u3$ to $u7$ are down-stream associated with CH, $u1$, $u2$, $X2$, and $X3$ respectively. In addition, we see that $X1$ and $X2$ are down-stream associated with $u2$ and $u1$ respectively, due to the initial malicious node list formed by CH.

After the above malicious association, the compromised CH can send any false reports to the sink without being detected en-route. The false report can be constructed through the collaboration between $X3$ and CH. In a false report, suppose P is the report content; $XMACP$ is the XOR-MAC to be authenticated by the sink [13]; the rest are five pairwise keyed MACs for en-route authentication. Let $key_{X2.u1}$ denote the pairwise key between node $X2$ and $u1$. The corresponding MAC – $\mathrm{MAC}_{key.X2.u1}(\mathrm{P})$ is generated by $X2$ and to be verified by $u1$. Clearly, if one node in an association pair is compromised, the MAC can always be forged. Let $MACx$ be some arbitrary unimportant bits in a MAC.

The report forged by CH is

$$[\ P, XAMCP,\quad MAC_{key.X2.u1}(P), MAC_{key.X1.u2}(P), MACx, MACx, MACx]$$

This false report can pass $u1$ and $u2$ since the corresponding MACs are generated by compromised nodes $X1$ and $X2$. After passing these two nodes, two new MACs are added – $MAC_{key.u1.u4}(P)$ and $MAC_{key.u2.u5}(P)$ which are to be validated by $u4$ and $u5$ respectively.

Therefore the compromised node $X3$ can rearrange the report and generate the new report as follows.

$$[\ P, XMACP,\quad MAC_{key.CH.u3}(P), MAC_{key.u1.u4}(P), MAC_{key.u2.u5}(P), MAC_{key.X2.u6}(P), MAC_{key.X3.u7}(P)]$$

In this report, the 1st, 4th and 5th MAC can be generated since each corresponding association has a compromised node. The 2nd and 3rd MACs are received from $u1$ and $u2$. At this point, all five MACs are consistent with the false report content. It can be forwarded to the sink without being detected.

Note that ACK messages are authenticated hop-by-hop in the original IHA scheme. However, this cannot defend the above attack. Since the node list in the ACK message gets updated at each node, the MAC gets updated as well. The MAC key used to authenticate the message is the pairwise key between the current node and its immediate up-stream node [13]. In the above attack, the malicious node deliberately changes the node list, and thus can always generate a consistent MAC to deceive the up-stream neighbor.

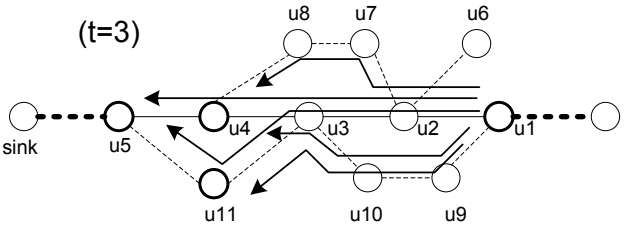
Attack summary. By observing this attack, we can find that node $X2$ sits on two authentication chains – the $u1$-chain and $u2$-chain. This effectively reduces the number of different MACs to t, or, it is possible to construct $(t+1)$ different MACs from t compromised nodes. In more general an attack succeeds if we can associate at least one compromised node on each of the $(t+1)$ authentication chains. We can also find that it is $X3$ that mainly perform the manipulation, resulting in that the attack can happen anywhere on the path. We will devise a mechanism to defend such attacks in section 4.

3.2 Problem 2: security and overhead tradeoffs in multipath routing

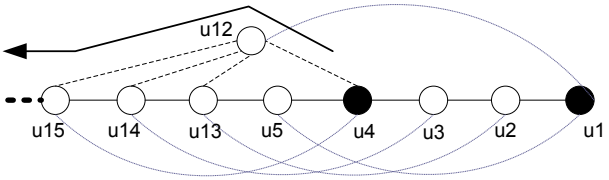
As false reports may also be injected in a multipath routing based sensor network, it is equally important to perform en-route authentication and drop false reports as early as possible.

However with braided multipath routing, there are large number of different routing paths between the sink and the source nodes. In [1] Ganesan *et al.* showed that the number is proportional to the n^{th} Fibonacci number if there are n nodes on the primary routing path. The maintenance overhead would be prohibitively high if the IHA scheme is directly applied to each possible path.

Instead a reasonable solution is to consider multiple routing paths simultaneously. To perform the node association at each node, only its neighboring nodes need to be considered. The IHA associates two nodes that are $(t+1)$ hops away from each other into one pair, and uses a node list containing the last traveled $(t+1)$ nodes to help node association. In multipath routing, we can similarly consider a short routing path segment. The difference is that there might be multiple $(t+1)$-hop-away nodes in each routing direction. It is not trivial to design a secure and efficient node association scheme accordingly.



(a) Multiple (t+1)-hop-away nodes



(b) The 3rd attempt: assign each into one association group

Fig. 6. The node association in multipath routing (t=3).

Let us discuss three attempts with different overhead and security levels. The first attempt is that a node sets up a different pairwise key with each of its $(t+1)$-hop-away nodes; authentication is then performed based on which path the report is routed along. For example Fig. 6(a) illustrates this attempt with t=3. There are five subpaths from $u1$ to $u5$,

$path_1$: $u1-u2-u3-u4-u5$

$path_2$: $u1-u2-u7-u8-u4-u5$

$path_3$: $u1-u9-u10-u3-u4-u5$

$path_4$: $u1-u9-u10-u3-u11-u5$

$path_5$: $u1-u2-u3-u11-u5$

Thus $u1$ creates three up-stream pairwise keys — one with $u5$ for $path_1$ and $path_5$, one with $u4$ for $path_2$ and $path_3$; and one with $u11$ for $path_4$. In addition, $u1$ has to create similar number of down-stream pairwise keys (not shown in the figure). This attempt is secure as each different subpath

is independently protected by IHA. However, as one can see, the overhead is still very high. In addition to the storage for saving and distinguishing multiple keys, a relay node (e.g. $u1$) has to generate and transmit three different MACs up-stream, instead of one MAC per node in IHA. The latter is more problematic as data transmission in sensor networks consumes more energy. The authentication overhead may increase m times in multiple routing if each node generates m MACs on average.

To reduce the authentication overhead, we can group all up-stream $(t+1)$-hop-away nodes together using one authentication key. Similarly, a different authentication key is created for all down-stream $(t+1)$-hop-away nodes. In the above example, a group authentication key is created among $u1$, $u5$, $u4$, and $u11$. Thus no matter which path the packet is to take, $u1$ only needs to generate one up-stream MAC. Unfortunately while this scheme can greatly reduce the overhead, the security is compromised. For example, due to the existence of $path_2$, node $u5$ also sets up a key with $u2$ as they are $(t+1)$ hops away from each other. Therefore, $u5$ stays on two different authentication chains ($u1$-chain and $u2$-chain). If $u5$ is compromised, it can generate two MACs such that $(t+1)$ legal MACs can be constructed by $u5$ and other $(t-1)$ compromised nodes. In summary, if a node gets involved in two different authentication chains, it is possible to break the authentication protection by generating $(t+1)$ legal MACs from t compromised nodes.

Based on the above observation, the third mechanism could be to allow each node to stay on only one authentication chain. In the above example, if $u5$ has been associated with $u1$, it may not be associated with $u2$ even with the existence of the path $path_2$. However, the difficulty is that between $u1$ and $u2$, which node should be associated with $u5$. We may run into security problems if we give the priority to one over the other. As another example, if the primary path is of higher priority in Fig. 6(b), then $u1$ to $u4$ should be associated with $u5, u13, u14, u15$ respectively. However, no matter how we associate node $u12$, only two different MACs are checked along the path $u4-u12-u15$. In such a setting even if a false report is detected and dropped along the primary path $u4-u5-u13-u14$, it might still reach $u15$ via the shortcut $u4-u12-u15$. If this continues to happen beyond $u15$, a false report can reach the sink without being dropped en-route.

Therefore, it is challenging to design an interleaved authentication scheme that is both secure and efficient for multi-path routing.

3.3 Assumptions

Before presenting our algorithm, we discuss the network and attack models that we consider in the paper.

The network model We consider a sensor network that consists of a number of battery-powered sensor nodes and a sink node with abundant resources, e.g. energy and computation power. We assume the sink node cannot be compromised. Each sensor is assigned with a unique ID and a secret key before deployment. Both the ID and the key are known to the sink node. Sensor nodes are left unattended after deployment. They monitor events of interests and send the data reports back to the sink. When an event happens in the network, it can be detected by multiple nodes in a cluster. We assume that majority of sensing nodes for any single event are trustworthy. We assume the clustering technique is used since it has been proven effective in reducing energy consumption of the entire sensor network [2,7]. Data reports are first sent to the cluster head who will construct an aggregated report that also contains the IDs and MACs from sensing sensors. We use the multi-hop braided multi-path routing scheme [1]. Thus the report is forwarded along both the primary and alternative routing paths to the sink.

The attack model We assume that once a sensor node is compromised, the adversary can retrieve all embedded security information including the secret key. Therefore, a compromised node can inject false data reports as shown in [13,10]. We further assume the adversary knows the protocol or other security algorithms used in the network.

We assume that the adversary can attack the node association phase. Since the node association may be performed periodically at the beginning of each epoch, some nodes may have already been compromised at that time. There are up to t compromised nodes in the network.

We assume the sink always has the ability to detect a false report as shown in [13]. It is achieved by including the XOR-MAC of MACs from $(t+1)$ sensing nodes using private keys. In this paper we therefore focus on detecting and dropping false report en-route.

4 Our Algorithm

In this section we present our algorithm for filtering false reports in multipath routing based sensor networks. We focus on enhancing node association schemes and then prove their security.

4.1 Overview

Similar to IHA, our algorithm contains five phases. The enhancements are integrated in phases 2 and 4 which will be discussed in more details next. Other phases stay unchanged.

1. The node initialization and deployment phase. Each node is loaded with a unique id and a private key before the deployment. The deployed node also sets up pairwise keys with its immediate neighbors.
2. The node association discovery phase. A node discovers its up-stream and down-stream associated nodes. Authentication keys are also generated in this phase.

3. The report endorsement phase. Each report is endorsed by $(t+1)$ nodes within the cluster. The cluster head collects the sensing data and the $(t+1)$ MACs, wraps them to one report, and sends the report back to the sink.
4. The en-route filtering phase. Each relay node verifies the MAC generated from its down-stream association nodes and generates *one* new MAC to be verified by its up-stream association nodes.
5. The sink verification phase. The sink node always has the ultimate ability to verify if the report is authentic using private keys of $(t+1)$ sensing nodes.

4.2 Detailed Description

We focus our discussion on node association since the authentication largely depends on how the nodes are associated. We first present how to defend the malicious node association attack in single path routing. We then extend it to the multipath based routing network.

Enforced node association for single path routing. Ideally IHA assigns $(t+1)$ consecutive nodes to $(t+1)$ different authentication chains. As there are at most t compromised nodes, even if t authentication chains are broken, at least one is still well-behaved. Any node on this chain can detect and drop false reports. However in Fig. 5 with the help of the compromised node $X3$ who forges the node lists for node association, node $X2$ is successfully linked to two different authentication chains. $X2$ is linked to the $u6$-chain through direct association while it is linked to the $u4$-chain through indirect association (through $u1$). Once the malicious node association succeeds, the adversary can defeat the authentication by generating $(t+1)$ consistent MACs from $X2$ and other $(t-1)$ compromised nodes. Since the node association is short-sighted to see past $(t+1)$ nodes, $u4$ and $u6$ do not know that their authentication chains actually merge at $X2$.

With the above observation, the enhancement is clear – we should find a way to enforce the interleaved node association such that $(t+1)$ authentication chains keep disjoint. A brute force approach is to remember the complete routing path from the cluster node to the sink. Clearly it is inefficient and expensive. Given there are at most t compromised nodes, we will prove it is secure by including last $(t+1)^2$ traveled nodes in the ACK message. Our enhancement works as follows,

- The ACK message contains nodes ids of last traveled $(t+1)^2$ down-stream nodes, that is, each node receives a matrix of nodes $[u_{ij}]$ $(0 \le i, j \le t)$). The matrix is to replace the $(t+1)$-node list in the ACK message in the original IHA. The nodes form $(t+1)$ authentication chain while each chain has nodes from last $(t+1)$ rounds. As shown in Fig. 7, given a fixed j $(0 \le j \le t)$, $u_{0j}, u_{1j}, ..., u_{tj}$ are on one authentication chain.

  In addition while $u_{00}, ..., u_{0t}$ are last traveled $(t+1)$ nodes, the last traveled one may not be u_{00}. Instead it is identified by the index value `Ind` as discussed next.

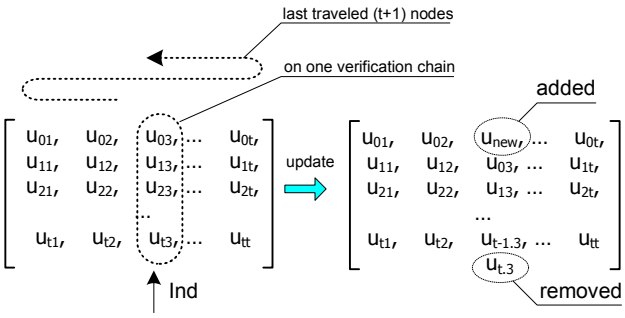


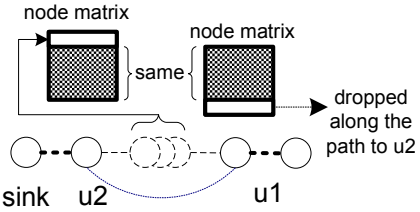
Fig. 7. An ACK message has $(t+1)^2$ node ids.

- An index value *Ind* with $\lceil log(t+1) \rceil$ bits is also included in the ACK message. It is used to indicate the last traveled node and its authentication chain. IHA identifies the last node by shifting the node to the head of the list. Instead we identify its column index in the matrix and keep updating the index. The reason is to help node association in multipath routing case.

  `Ind` is initialized to zero; when a node receives the node list, it is updated as follows.

$$Ind_{new} = (Ind_{received} + 1) mod (t+1).$$

  After updating `Ind`, the current node is associated to the $u_{0.Ind}$ authentication chain.
- Each node also updates the id matrix before sending to the next up-stream node. We do not shift the whole matrix but instead only the one column, that is, the current node is set as $u_{0.Ind}$ while $u_{k.Ind}$ replaces $u_{(k+1).Ind}$ $(0 \le k \le$ t-1)). Node $u_{t.Ind}$ is removed from the list (Fig. 7).
- To prevent node id matrix from malicious modification, the authentication key is generated using the node matrix and the index. That is, instead of setting the authentication key as the pairwise key `PairKey` between two directly associated nodes, we generate a new authentication key `AuthKey` from the pairwise key, the index `Ind` and the common part of the node matrices at two nodes.



As we know any two associated nodes should have the same `Ind`. After routing a node matrix to its up-stream associated node, the last $(t+1)$ node ids should have been removed, the first $(t+1)$ nodes are newly added, and the rest are shifted by one row. The newly added node ids are those ids traveled between $u1$ and $u2$. As a result, $u1$ and $u2$ can create a common authentication key only if they are on the same authentication chain and see the same $t(t+1)$ traveled nodes before $u1$.

To generate the consistent authentication key, we have (note u_{ij} at $u1$ is the same as $u_{(i+1)j}$ at $u2$)

at $u1$: $AuthKey = F_{PairKey}(Ind||u_{00}||u_{01}||...||u_{(t-1)t})$.

at $u2$: $AuthKey = F_{PairKey}(Ind||u_{10}||u_{11}||...||u_{tt})$.

where F is an encryption or a keyed-hashing function, e.g. RC5 [10]; `PairKey` is the pairwise key shared between $u1$ and $u2$. $u1$ and $u2$ must be sharing the same `Ind` as they are on the same authentication chain.

- Before generating the new authentication key, the current node checks nodes from any two authentication chains do not overlap, i.e. no node can appear twice in the node id matrix attached in the ACK message $\forall u_{ij}, u_{kl}, u_{ij} = u_{kl} \;\; iff \;\; (i = k \; and \; j = l)$.

We will prove that the above enhancement ensures that any $(t+1)$ consecutive non-compromised nodes stays on disjoint authentication chains. Now the attack in Fig. 5 cannot succeed: $u4$ will reject the association as $X2$ appears on the $u4$-chain and it is one of the last traveled t nodes.

Node association for multipath routing. Next we discuss how to achieve secure and efficient authentication in multipath routing. We split the node association to two subtasks and want to achieve

(1) The up-stream association: to maintain similar authentication overhead as that in single path routing, each node only generates one MAC regardless of the path a report may further be forwarded to. In other words, a node shares the same authentication key with multiple up-stream nodes.
(2) The down-stream association: to ensure the authentication strengthen, i.e. no authentication chain may be skipped, we perform conservative down-stream node association assuming the path with the smallest number of hops (and authentication checks) was taken. That is, the authentication is interleaved according to the shortest path while it may be out of the interleaving order along other paths.

Next we use an example to illustrate how our scheme works and then present the detailed algorithm. In Fig. 8(a), we assume t=3 and there are four $v1, v2, v3, v4$ authentication chains. Let us first focus on the down-stream node association at node $u5$. Assuming that nodes $u1$ to $u4$ have been interleavingly associated to $v1$- to $v4$- chains respectively. node $u7$, $u10$ and $u11$ have been associated to $v3$-, $v4$-, and $v1$- chains respectively. According to two different paths $u1-u2-u3-u4-u5$ and $u1-u2-u3-u10-u11$, $u5$ can be associated either to $v1$-chain or $v2$-chain. We want to associate it to one chain to ensure security (otherwise t compromised nodes may generate $(t+1)$ consistent MACs). With the conservative association policy, $u5$ is associated on $v1$-chain, which ensures that checks on $v1$-chain are not missed along any path.

The above association leaves the authentication on the path $u1-u2-u3-u10-u11$ not strictly interleaved. More importantly, along this path, the association between $u5$ and

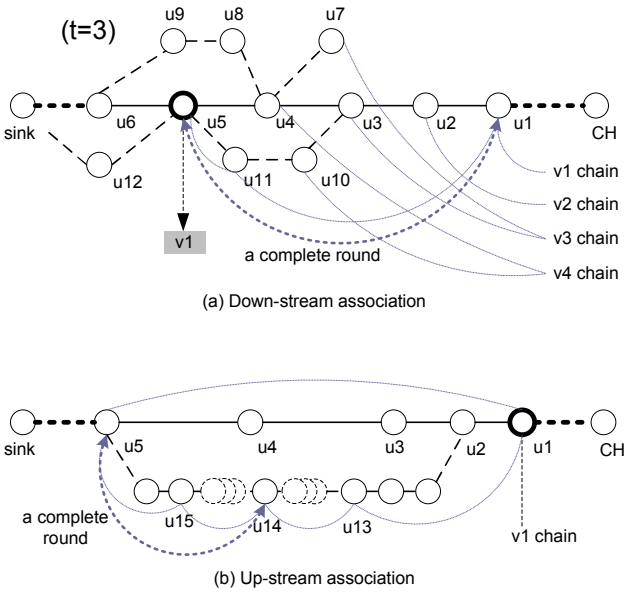


Fig. 8. The conservative node association in multipath routing (t=3).

$u11$ has no other authentication chains in between. We consider it as an *incomplete* round and define

Definition 5 *Given a compromised node threshold t, there are $(t+1)$ authentication chains. A node V_2 is defined as one* complete round *away from V_1 if (a) V_1 and V_2 are on the same authentication chain, and along the path segment from V_2 to V_1 we can find nodes from all other t authentication chains; (b) there is no node V_3 on the same authentication chain between $V1$ and $V2$ such that along the path segment from V_3 to V_2 we can find nodes from other t authentication chains.*

Definition 6 *If V_2 and V_1 are on the same authentication chain, and along the path segment from V_2 to V_1 there misses nodes from at least one of other t chains, then V_2 is a* dangling *node to V_1.*

For example, in Fig. 8(a), $u1$ is one complete round away from $u5$ while it is two (raw) rounds away from $u5$. And $u11$ is a dangling node to $v5$. With these concepts, the ACK message in multipath routing will includes nodes from last $(t+1)$ complete rounds (including all dangling nodes) for each of $(t+1)$ chains. We will use the property that a complete round contains all $(t+1)$ authentication chains to prove the correctness of our scheme.

Let us then look at the up-stream association at the node $u1$ in Fig. 8(b). it has two up-stream nodes $u5$ and $u13$ both of which are four hops away. Since we only want to generate one MAC at node $u1$, nodes $u1$, $u5$, $u13$ are grouped to

share one authentication key. On the other hand, a node does not have to share the same authentication key with all its down-stream nodes. That is, $u13$ may keep two downstream authentication keys, one (with $u1$) for the packet routed along $u4-u5$ and the other one (with $u15$) for the packet routed along $u15-u5$.

We should be careful regarding the group authentication key generation. That is, assuming node $u13$ is a compromised node, it may try to associate with $u4$ as well. If it succeeds, $u13$ can stay on two different chains. To prevent this, node $u4$ has to know that $u13$ has already been on $u1$-chain. Since $u1$ does not have the information beforehand, our scheme sends one extra message just to find the association relationship but leaves the actual key generation in processing the ACK message.

The algorithm details are as follows.

- The algorithm contains three steps. In the first step, the CH (cluster head) sends a SYN message to the sink along all braided paths. This is to decide on which authentication chain each of the relay node should be assigned to. In the second step, the sink sends a NOTIFY message to the CH. Each node gets a notification from each of its up-stream associated nodes. In the third step, the CH sends an ACK message with the related node list to generate the authentication key.
- In step 1, the SYN message contains an index `Ind` and a value `Rcnt` that counts the number of association rounds.

  If a node u_x receives one SYN message from u_y, it identifies itself as the next node of u_y. If u_x receives more than one SYN messages, it identifies itself as the next node along the path with the smallest `Rcnt`, or the smallest `Ind` if all `Rcnt` are the same. u_x updates both `Ind` and `Rcnt` before forwarding them further.

  For example in Fig. 8(a), $u5$ and $u6$ are assigned to v1- and v2- chains respectively.
- In step 2, the NOTIFY message contains up to $(t+1)$ sub-lists. Each sub-list $L_i(0 \le i \le t)$ corresponds to one authentication chain and contains all traveled but unassociated nodes on that chain i, i.e. they have not been associated with downstream nodes. Clearly a sub-list may contain multiple such nodes along different paths. The NOTIFY message also includes an index field that helps to identify the incomplete round. The steps for processing the NOTIFY message are in Fig. 9.

  Each node may receive multiple NOTIFY messages. The current node is associated to all nodes with the same index in these messages (a group key will be generated for these nodes in step 3). We then check dangling nodes by checking if two consecutive nodes are not on consecutive chains. The current node gets associated with its upstream same-chain nodes if no dangling is found; otherwise, it is simply added to the corresponding sub-list and gets associated later on (Line (2)-(9)).

```
Assume the current node is Nx with the index Indx

(1)  for each NOTIFY message with node list (L0,L1,...,Lt) and
     the expected chain index Eind from the upstream node Ny
(2)     if (Eind == Indx)
(3)         associate Nx with all nodes in LIndx;
(4)         Eind = (Eind + t) % (t+1);
(5)         LIndx = {Nx};
(6)     else
(7)         mark Nx as a dangling node to LIndx;
(8)         LIndx = LIndx ∪{Nx};
(9)     endif
(10) endfor
(11) merge lists (with same Eind) from all NOTIFY messages.
(12) send updated messages to Nx 's one-hop downstream nodes.
```

Fig. 9. The algorithm for processing the NOTIFY message.

For example in Fig. 8(a), dangling is found along path $u5$-$u11$ as they are of the same chain index. Therefore in the outgoing NOTIFY message from $u11$, the v1-chain sub-list contains both $u5$ and $u11$. As another example, in Fig. 8(b), no dangling is found along $u2$-$u1$ such that $u1$ is notified that both $u5$ and $u13$ are to associate with it.

- In step 3, an ACK message is sent for generating the authentication keys. The ACK message includes nodes from last $(t+1)$ complete rounds of each $(t+1)$ authentication chains. Due to multiple ACK messages received by each node, each chain is processed individually. The algorithm steps are shown in Fig. 10.

```
Assume the current node is Nx with the index Indx
Ltmp contains Nx's directly associated downstream nodes // from step2

(1)  for each incoming ACK message with lists Lij (0≤i,j≤t)
(2)      if any node appears on two different columns (i.e. chains)
(3)          alarm "association attack";
(4)      endif
(5)  endfor
(6)  if (Nx is a dangling node)
(7)      return;
(8)  endif
(9)  for all ACK incoming messages
(10)     Nz = first node in L0.Indx;
(11)     generate a group downstream key with Nz (using L0.Indx);
(12)     for (0≤i≤t-1)
(13)         L(i+1).Indx = Li.Indx
(14)     endfor
(15)     L0.Indx= Nx ∪ Ltmp;
(16) endfor
(17) merge ACK messages with the same Lij
(18) generate a group upstream key.
```

Fig. 10. The algorithm for processing the ACK message.

A node N_i may receive multiple ACK messages. Each message contains a $(t+1)\times(t+1)$ matrix L_{ij} to memorize

nodes from last $(t+1)$ complete rounds on each authentication chain j ($0 \leq$ i,j$\leq$t). An item in L_{ij} represents the related nodes for round j of the authentication chain i. The first node of each $L_{0.j}$ is reserved for the latest traveled node on authentication chain j. The ACK message also includes a $(t+1)$-item integer array indicating if dangling is found in the last round of each authentication chain.

Lines (1)-(5) check if a node appears on two different chains and alarm the attack if such a node is found. Next we process the incoming ACK messages at node Nx. If it is not a dangling node, Nx gets associated with Nz, the last traveled node on the same chain. In generating this downstream group key, we also need to use $L_{0.Indx}$ which contains all associated upstream nodes of Nz (line (11)). The corresponding column of L_{ij} is then shifted to reflect the new node association. Nx also generates a group upstream key that is shared with its upstream associated nodes (line (18)).

- To generate the authentication key in step 3. All nodes in one association group, e.g. $u1$, $u5$, and $u13$ in Fig. 8(b), compute from `Rcnt`, `Ind` and the shared node ids in their received ACK messages. That is $AuthKey = F_S(Rcnt||Ind||Shared\_ids)$. Where S is a secret selected by $u1$ and transferred to its up-stream associated nodes using the pairwise keys.

  In generating the key, a downstream node e.g. $u1$ identifies all shared nodes from [L_{ij}] ($1 \leq$ i$\leq$t,$0 \leq$ i$\leq$t) at $u1$ while a upstream $u5$ finds these nodes from [L_{ij}] ($0 \leq$ i$\leq$(t-1),$0 \leq$ j$\leq$t) at $u5$.

En-route message authentication. After the node association, the en-route message authentication works similar to that in single path authentication. Each report contains $(t+1)$ MACs, each relay node picks up the one to verify according to its `Ind` and the authentication key created in the node association phase. For example in Fig. 8(a) $u5$ always picks up the third MAC to verify as it was assigned to $v3$.

After the authentication, a new MAC is generated using the authentication key for its up-stream association nodes and replaces the old one in the report. The report is then forwarded following multiple outgoing paths.

The only difference is that, since a report may take different paths to reach a merge node, the MAC of the same index may be different along different paths. For example in Fig. 8(b) we may use either $AuthKey_{u5 \cdot u14}$ or $AuthKey_{u1 \cdot u5 \cdot u13}$. We can either include several extra bits to distinguish or try all downstream keys. Our results show either approach introduces negligible overhead.

4.3 Security Analysis

In this section we analyze the security of our enhancements. In particular we prove that the proposed scheme can defend the malicious node association attack in both single path and multipath routing based networks.

Theorem 1 *The proposed scheme can defend node association manipulation attack in single path routing sensor networks.*

Proof We will prove by contradiction. Let us assume that a false report can escape the authentication check, that is, it can be routed through consecutive $(t+1)$ *non-compromised* nodes without being detected.

Let us denote the first such routing path segment as SEG_1. It contains $(t+1)$ non-compromised nodes $v_1, v_2, ...v_{(t+1)}$. The t compromised nodes are $X_1, X_2, ..., X_t$.

Since there are $(t+1)$ authentication chains and t compromised nodes, we can find at least one compromised node that stays on two authentication chains. Otherwise we will have an authentication chain that consists of non-compromised nodes. The false report will be detected if any node on this chain receives the report.

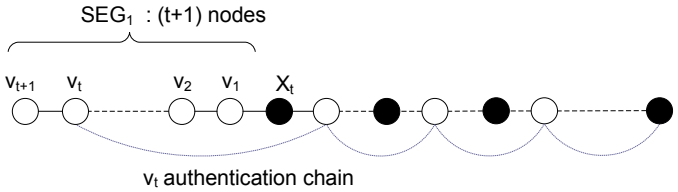


Fig. 11. V_t has to meet a compromised node with t association rounds.

The node before reaching v_1 must be a compromised node, otherwise the first non-compromised $(t+1)$-node path segment would be from this node to v_t. Accordingly there is a compromised node between v_t and its down-stream associated node v_{dt}. Similarly we can find a compromised node between v_{dt} and its down-stream associated node, and so on. Given there are at most t compromised nodes, this process can continue at most t rounds, or a compromised node is met on the chain. If it jumps t rounds, then all t compromised nodes have been jumped and therefore it is an authentication chain without any compromised node. as we discussed above this chain can detect a false report immediately.

In other words, all $(t+1)$ authentication chains must meet a compromised node within t rounds of association down-stream, i.e. with $(t+1)^2$ nodes down-stream. A compromised node X_1 therefore has to stay on two chains e.g. v_1 and v_2 chains. From the algorithm, at least $t(t+1)$ shared nodes received by $v1, ..., v_{(t+1)}$ cannot be compromised. Otherwise different lists are used to generate keys such that the consistent authentication key cannot be reached between associated nodes. On the other hand if a node stays on two chains with t rounds it is immediately identified in the algorithm such that the node association cannot succeed.

To summarize, it is impossible to associate one compromised node with two authentication chains. Accordingly a

false report cannot be routed through consecutive $(t+1)$ *non-compromised* nodes without being detected. The node association attack is defended. □

Theorem 2 *The proposed scheme can defend node association manipulation attack in multipath routing based sensor networks.*

Proof To prove in multipath routing networks, we follow the similar strategy as that in the above proof.

If a false report can escape the authentication check, it has to route through a routing path segment that contains nodes from $(t+1)$ different authentication chains. We assume the first such segment is SEG_1. It may have more than $(t+1)$ nodes since node association is not strictly interleaved along some paths. Due to the fact that there are at most t compromised nodes, we can find at least one compromised who can generate two MACs that can be accepted into two authentication chains. This means this compromised node should share the authentication keys with some nodes on two different authentication chains.

In the node association scheme we designed for multipath routing, a compromised node X has two choices to stay on an authentication chain and thus share an authentication key with other nodes. Regarding the $u5$-chain in Fig. 8(b), X can either be $u1$ which is a $u5$-chain node and on the routing from $u5$ to CH, or $u13$ which is a $u1$-chain node but not $u5$-chain node. $u13$ is directly associated with a node $u1$ on the $u5$-chain. We denote all these nodes as *related* nodes in the algorithm and include them in the matrix. Therefore, we can conclude that a compromised node has to be one of such related nodes on two different authentication chains.

We next prove that all compromised t nodes must be within t downstream complete rounds from SEG_1. Since our algorithm shifts the authentication chain only if the last traveled node on the same chain is not a dangling node, therefore the $(t+1)$ items on one chains remembers at least $(t+1)$ complete rounds. In addition, all *related* nodes from these downstream rounds are included. Therefore no matter how we are going to distribute the t compromised nodes, we can find another path segment before SEG_1 that have $(t+1)$ different authentication chains.

Since we check node overlapping before generating the authentication key, it is impossible to let one compromised node break into two chains without being detected.

Therefore we should have at least one authentication chain which contains no compromised node. The false report will be detected and dropped immediately if any node on this chain receives the packet. As nodes are associated conservatively according to the shortest path, it is impossible to skip such an authentication chain.

In summary, we can defend the node association attack in the multipath routing. □

4.4 The packet size overhead

Our scheme follows the design philosophy of interleaved authentication approaches: each report is attached with $(t+1)$ MACs for authentication purpose [13,18]. As discussed in recent work [18], when the overhead becomes a concern due to large t, an alternative design is to attach only a portion of each MAC. The MAC computation is still 64-bit based such that there is no compromise of algorithm security. The overhead is greatly reduced due to fewer bits attached to each report, and at the cost of some space left to the attacker to guess with higher success rate. For example, if first 16 bits of each MAC are used, a random guess would have $1/2^{16}$ instead of $1/2^{64}$ probability to succeed. This brings a performance degradation but not a security problem i.e. few false reports may travel more hops and reach the sink. False reports can still be detected from the sender MAC and thus no false negative. We evaluated the performance in the next section.

5 Experiments

5.1 Settings

We have implemented our algorithm and simulated it in a network with 1000 sensors deployed in a 1000×250 m^2 region. Each sensor node has the detection range of 20m. The aggregation nodes and the sink node are set at two opposite ends of the region with about 50 routing hops in between. Each sensor node is similar to Mica2 operating at 19.2Kbps data rate, with battery voltage 3V. It takes 16.25/12.5 μJ to transmit/receive a byte [10]. We ignore the computation energy cost which is usually small comparing to routing energy consumption. In the simulation, to isolate and illustrate the impacts due to applying different authentication algorithms, we did not consider node interference and channel contention as these factors depend more on the environment condition, the time interval between consecutive data collection operations, the amount of data to collect, and the time constraint in receiving the reports. Channel contention and packet loss due to these factors negatively impact all algorithms.

We assume that the adversaries have the full control of the compromised nodes. They may selectively stay anywhere on the authentication chain, inject false reports, and collaboratively generate consistent MACs to deceive the en-route authentication. That is we evaluate the worst case performance of discussed algorithms. In case if all MACs match the false content, the false packet can be routed all the way to the sink node where it then gets detected by the sink node. Legitimate sensors behave according to the given authentication algorithms.

5.2 The node association overhead

Fig. 12 compares the overhead to setup the node association in different schemes. Since all schemes set up the as-

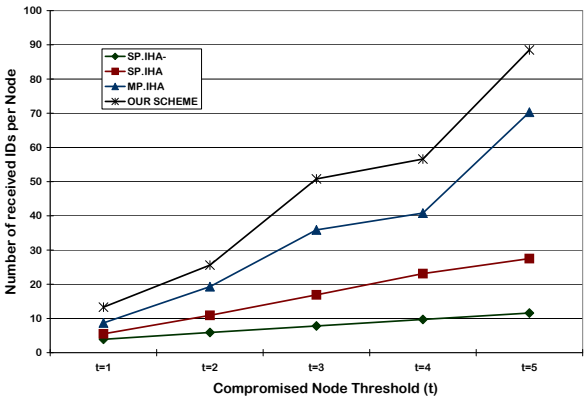


Fig. 12. The node association overhead.

sociation by exchanging messages that contain neighboring node IDs, we represent the overhead as the number of IDs each relay node receives. In the original IHA [13] for single path routing — denoted as `SP.IHA-`, each node receives two messages while each contains last traveled $(t+1)$ IDs. Therefore its overhead is 2*$(t+1)$/node. `SP.IHA` represents the scheme with the enhancement to defend the association attack. In this scheme, each ACK message contains $(t+1)^2$ node IDs and thus the overhead is $((t+1)^2+(t+1))$/node. The results are slightly smaller than the theoretic data because nodes close to CH and the sink receive messages with less number of IDs. For example, in `SP.IHA` the ACK message received by nodes within t^2 hops away from the CH contain less than $(t+1)^2$ IDs.

To perform the interleaved authentication in multipath routing, a naive adoption of the IHA scheme, denoted as `MP.IHA`, needs to create association with multiple upstream and downstream nodes. It has higher overhead than `SP.IHA` as a node should receive nodes IDs from all paths e.g. the ACK message contains last traveled $(t+1)^2$ from all possible paths. From the figure, our new scheme has the highest setup overhead. This is because that our scheme exchanges three messages. The SYN message contains two values; the NOTIFY message contains a node list with O(t+1) node ids; the ACK messages contains a node list with O(k$\times(t+1)^2$) node ids where k is the average number of different ACK messages per node. This is similar to `MP.IHA`, the difference is that due to the selection of the path with the smallest number of hops, a complete round path segment may have more than $(t+1)$ nodes along some paths. On average in association phase, our scheme receives 38% more node IDs than `MP.IHA`.

While the overhead of the node association in our scheme is higher, it is a one-time overhead in each epoch and quickly amortized by multiple relayed packets.

Other cost includes the several bits in the report to distinguish the $(t+1)$-hop routing path such that a relay node can select from multiple down-stream keys. The required number of bits is in very small and can be removed if there is only one down-stream associated node.

5.3 *The routing overhead*

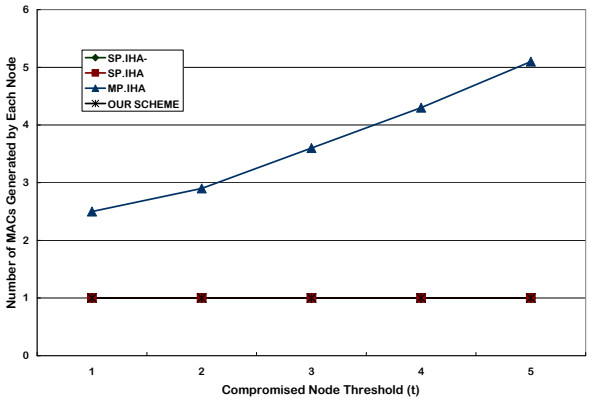


Fig. 13. The number of MACs generated by each node.

Next we study the routing overhead. Fig. 13 shows the number of MACs generated at each node. Clearly in single path routing schemes, each node needs to verify and replace one MAC i.e. only one new MAC is generated. In `MP.IHA`, since each node is associated with multiple upstream nodes, a node has to generate multiple MACs. Depending on which path the report is further forwarded, a corresponding MAC might be used to verify the report. For example, on average MP.IHA generates 3.6 MACs per node when the threshold t is set to 3. On the other hand, our scheme always generates one group authentication key for all upstream nodes. Therefore only one MAC is generated at each node – the same as that in the single path routing.

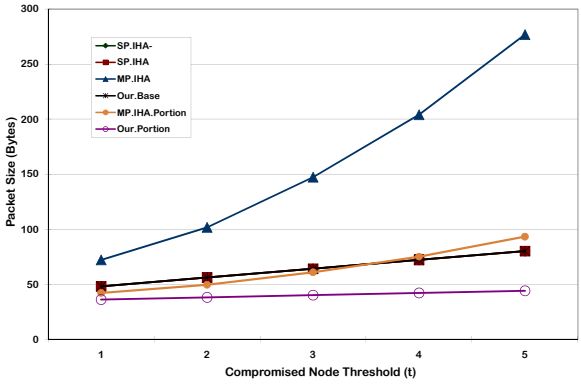


Fig. 14. The packet size in different schemes.

Fig. 14 illustrates the the average packet sizes for different schemes. We first evaluate when the portion of each MAC is attached to the packet. With more MACs generated and included into the report at each node, the report size increases rapidly, which results in more energy consumption. For example, when t=4, a node in `MP.IHA` generates 4.3 MACs on average, i.e. a report on average contains 21.5 (=4.3$\times(t+1)$) MACs. Given the fact that the MAC size is 8 bytes and a report content is of 24 bytes, the authentication overhead (the total size of all MACs) is about 8 times larger than that of the content. A large packet is also undesired as it increases the chance of channel contention. On the other hand, our

scheme includes 5 MACs. It is the same as that in the single path routing, and a 64.7% reduction from `MP.IHA`.

If we attach a portion of of each MAC as discussed in section 4.4 and [18], then per-packet size overhead can be greatly reduced. As shown in the figure, `MP.IHA.Portion` has a 63.2% reduction from `MP.IHA`. Our scheme gains additional 41.7% reduction from `MP.IHA.portion` when t=4. In this case, the total MAC bits for interleaved authentication are 80 bits, or a 31% increase from the packet size without en-route authentication.

5.4 *False report dropping*

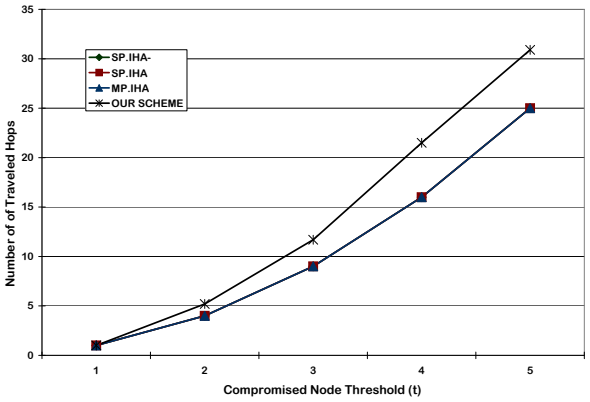


Fig. 15. The number of hops that a report travels before being dropped (the worse case).

We then study the effectiveness of our scheme in dropping injected false reports. Since interleaved authentication schemes are $(t+1)$ resilient, in the worst case there are t compromised nodes that are separated evenly along the routing path — there are t different authentication chains between any two compromised nodes. Fig. 15 summarizes the average number of hops that an injected false report can travel before it is being detected and dropped. As we can see, a false report travel more hops in our schemes. This is because in our scheme a one-round path segment that has $(t+1)$ different authentication chains may contain more than $(t+1)$ nodes. On average a false report travels 23.6% more hops in our scheme.

With the above analysis, we then compare the energy consumption of injected false reports in `MP.IHA` and our scheme (Fig. 16). Regarding the wasted energy of each injected false report, we have the following formula

$$Energy_{wasted} = Rsize \times Hcnt \times (E_{transmit} + E_{receive})$$

where *Rsize* is the report size, *Hcnt* is the number of hops that a false report travels before being dropped, and $E_{transmit}$ *and* $E_{receive}$ are the energy consumed to transmit and receive the false report at each node.

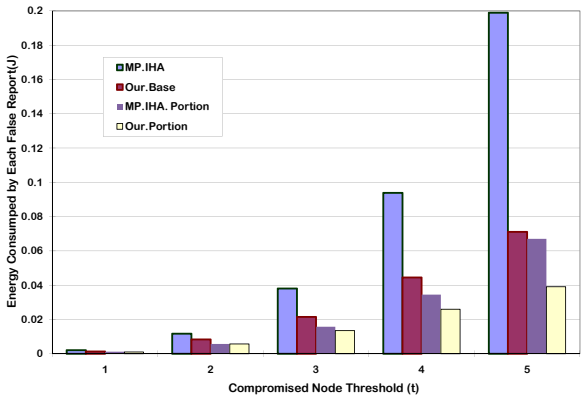


Fig. 16. The energy consumption per false report (the worse case).

Comparing to `MP.IHA`, a false report travels more hops in our scheme, however the report is much smaller as it contains less number of MACs. In summary our scheme greatly reduces the energy consumption of injected false reports. For example, when t=5, we save 64.3% of the energy with the base design.

If a portion of MAC is used, then the routing energy overhead can be greatly reduced. For example, `MP.IHA.protion` uses 16 bits and has a 66.3% reduction from the base line using `MP.IHA` when t=5. The reduction is also achieved in our scheme. However, if for example 16 bits are used and the attack random guesses the bits, then 1 out of 65536 reports escapes the check. Considering this extra overhead, we draw the overall energy consumption using our scheme. The reduction is 41.6% from `MP.IHA.Portion`. While the percentage is less than that in the baseline comparison, it is still significant.

The impacts on the lifetime of the network from different algorithms are proportional to the wasted energy per false report (Figure 16). The more energy each false report wastes, the shorter lifetime the network has. The exception is, if a false report can bypass the en-route detection i.e. due to insecure authentication algorithms e.g. *SP.IHA*- and *MP.IHA*-, then it can travel up to the longest routing path in the network and can only be detected and dropped by the sink node. If the attacker keeps injecting false reports along this path, then its impact on network lifetime is unbounded (assuming no other schemes are employed).

5.5 *Comparison with public key based authentication*

We then compare our scheme with the public key authentication scheme ECC [16]. We select ECC rather than RSA since ECC has shorter key length and tends to be more communication-energy-efficient. Recent study shows that with public key authentication, the computation dominates the total overhead — it take about 4 to 5 seconds to sign and verify on MICAz node using 128-bit ECC, or about 100mJ [17]. It is still too expensive to adapt to hop-by-hop based authentication schemes such as IHA. If a shorter key is used

in ECC, then the security is severely impacted. It is not like the adoption of MAC portion in section 4.4. Once a public key with shorter bits is compromised, all false packets using this key cannot be detected. In section 4.4, the MAC is still generated from secret symmetric key, only a small number of false reports (e.g. 1 out of $1/2^{16}$) can escape the en-route check.

In addition, ECC needs to store the public keys of other nodes. A sensor node can either store the keys from all other nodes, or from selected neighbors. The former is impractical due to its prohibitive storage request. The latter (storing a subset of keys) is also problematic since a relay node may not have the key to authenticate the received packet.

6 Conclusions

In this paper we studied en-route false report filtering in multipath routing based sensor networks. We identified the node association manipulation attack and the association problems in multipath routing. We proposed schemes to achieve secure and efficient authentication, and analyzed their security and performance. The schemes achieve similar en-route authentication overhead and filtering upper bound as these in single path routing.

Acknowledgment

This work was supported partially by NSF CAREER grant CCF 0641177 and NSF grants CCF 0430021, CNS 0720595.

References

[1] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-resilient, Energy-efficient Multipath Routing in Wireless Sensor Networks," In *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol.5(4), 2001.

[2] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Transactions on Wireless Communications*, vol 1:4, pages 660-670, 2002.

[3] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: a Scalable and Robust Communication in Wireless Sensor Networks," In *5th IEEE/ACM Mobicom*, pages 174-185, 1999.

[4] C. Karlof and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures," in *IEEE International workshop on Sensor Network Protocols and Applications*, pages 113-127, 2003.

[5] J.N. Al-Karaki, and A.E. Kamal, "Routing Techniques in Wireless Sensor Networks: A Survey," in *IEEE Wireless Communications*.

[6] S. Dulman, T. Nieberg, J. Wu, and P. Havinga, "Trade-off between Traffic Overhead and Reliability in Multipath Routing for Wireless Sensor Networks," In *WCNC workshop*, 2003.

[7] V. Kawadia and P. R. Kumar, "Power Control and Clustering in Ad Hoc Networks," In *INFOCOM*, 2003.

[8] Y. Xu, J. Heidemann, D. Estrin, "Geography-informed Energy Conservation for Ad Hoc Routing," in *ACM MOBICOM*, 2001.

[9] H. Yang, F. Ye, Y. Yuan, S. Lu and W. Arbaugh, "Toward Resilient Security in Wireless Sensor Networks," In *ACM MOBIHOC'05*, 2005.

[10] F. Ye, H. Luo, S. Lu and L. Zhang, "Statistical En-route Detection and Filtering of Injected False Data in Sensor Networks," In *IEEE INFOCOM 2004*, 2004.

[11] F. Ye, G. Zhong, S. Lu, and L. Zhang, "GRAdient broadcast: A robust data delivery protocol for large scale sensor networks," In *ACM Wireless Netw. (WINET)*, vol. 11, no. 2, Mar. 2005.

[12] W. Zhang and G. Cao, "Group Rekeying for Filtering False Data in Sensor Networks: A Predistribution and Local Collaboration-Based Approach," In *INFOCOM*, 2005.

[13] S. Zhu, S. Setia, S. Jajodia, P. Ning, "An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks," In *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, California, May 2004.

[14] B. Bloom, "Space/time Tradeoffs in Hashing Coding with Allocable Errors," Communication of ACM, Vol. 13, no. 7, pages 422-426, 1970.

[15] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary Cache: a Scalable Wide-area Web Cache Sharing Protocol," In IEEE/ACM Transactions on Networking, Vol 8(3):281-293, 2000.

[16] D. Hankerson, A. Menezes, S. Vanstone, "Guide to Elliptic Curve Cryptography," Springer Verlag New York, Inc. 2004.

[17] P. Ning, and A. Liu, "TinyECC: Elliptic Curve Cryptography for Sensor Networks,", http://discovery.csc.ncsu.edu/software/TinyECC/.

[18] Kui Ren, Wenjing Lou, and Yanchao Zhang, "LEDS: Providing Location-aware End-to-end Data Security in Wireless Sensor Networks," *Infocom*, 2006.