University of Colorado, Department of Computer Science Technical Report CU-CS-939-02

INSENS: Intrusion-Tolerant Routing in Wireless Sensor Networks

Jing Deng, Richard Han, Shivakant Mishra

Department of Computer Science
University of Colorado, Boulder, CO 80309-0430.
Contact: {rhan, mishras}@cs.colorado.edu

ABSTRACT

This paper describes an INtrusion-tolerant routing protocol for wireless SEnsor NetworkS (INSENS). INSENS constructs forwarding tables at each node to facilitate communication between sensor nodes and a base station. It minimizes computation, communication, storage, and bandwidth requirements at the sensor nodes at the expense of increased computation, communication, storage, and bandwidth requirements at the base station. INSENS does not rely on detecting intrusions, but rather tolerates intrusions by bypassing the malicious nodes. An important property of INSENS is that while a malicious node may be able to compromise a small number of nodes in its vicinity, it cannot cause widespread damage in the network. A prototype implementation in the ns2click simulator is presented to demonstrate and assess INSENS's tolerance to malicious attacks launched by intruder nodes in random and grid topologies.

Keywords: Wireless sensor networks, multi-path routing, intrusion tolerance, security, resource constraints.

1 INTRODUCTION

Wireless sensor networks (WSNs) are rapidly emerging as an important new area in mobile computing research. Applications of WSNs are numerous and growing, and range from indoor deployment scenarios in the home and office to outdoor deployment scenarios in natural, military and embedded environments. For military environments, dispersal of WSNs into an adversary's territory enables the detection and tracking of enemy soldiers and vehicles. For home/office environments, indoor sensor networks offer the ability to monitor the health of the elderly and to detect intruders via a wireless home security system.

In each of these scenarios, lives and livelihoods may depend on the timeliness and correctness of the sensor data obtained from dispersed sensor nodes. As a result, such WSNs must be secured to prevent an intruder from obstructing the delivery of correct sensor data and from forging sensor data. To address the latter problem, end-to-end data integrity checksums and post-processing of sensor data can be used to identify forged sensor data. This paper focuses on the former problem and develops a secure routing system to address the issue of obstructing packet delivery, which is an acute problem in sensor networks since each individual node can be easily compromised and thereby lead to the entire sensor network being compromised.

The design and implementation of secure WSNs must simultaneously address several difficult research challenges. First, wireless communication among the sensor nodes increases the vulnerability of the network to eavesdropping, unauthorized access, spoofing, replay and denial-of-service (DOS) attacks. Second, the sensor nodes themselves are highly resource-constrained in terms of limited memory, CPU, communication bandwidth, and especially battery life. These resource constraints limit the degree of encryption, decryption, and authentication that can be implemented on individual sensor nodes, and call into question the suitability of traditional security mechanisms such as compute-intensive public-key cryptography for such resource-constrained sensor nodes. Third, WSNs face the added physical security risk of individual sensor nodes falling into the wrong hands. Sensor nodes that are physically deployed in the field can be obtained by an intruder, and can then be subject to attacks from the potentially well-equipped intruder in order to compromise a single resource-poor node. Following a successful attack, a compromised sensor node could then be used to instigate such malicious activities as advertising false routing information, possibly unbeknownst to the sensor network, and launching DOS attacks from within the sensor network.

The combined threats introduced by increased physical security risk and severe resource constraints motivate the following design philosophy to achieve secure WSNs: concede that a well-equipped intruder can compromise individual sensor nodes, but secure the overall design of the WSN so that these intrusions can be tolerated and the network as a whole remains functioning despite such localized intrusions. More precisely, our objective is the design of *intrusion-tolerant* secure WSNs that have the property that a single compromised node can only disrupt a localized portion of the network, and cannot bring down the entire sensor network. This design objective of intrusion tolerance for secure WSNs must provide protection against two classes of attack that could bring down an entire sensor network: DOS-type attacks that flood data packets to the entire network; and routing attacks that propagate erroneous control packets containing false routing information throughout the network.

Intrusion tolerance can be designed to take advantage of a common architecture found in WSNs, namely the asymmetric architecture pictured in Figure 1. A base station functions as a gateway, e.g. an uplink to a satellite or a bridge between the wireless world of the WSN and the wired infrastructure seeking to process and mine the sensor data. Such a base station typically has more resources in terms of power, computation, memory, and bandwidth than the individual sensor nodes.

To achieve intrusion tolerance given an asymmetric topology and resource constraints, this paper presents an INtrusion-tolerant routing protocol for wireless SEnsor NetworkS (INSENS). The INSENS secure routing system adheres to the following high-level design principles. First, to prevent DOS-style flooding attacks, the type of communication is constrained. Individual nodes are not allowed to broadcast to the entire network. Only the base station is allowed to broadcast. We describe later on how authentication of the base station is achieved via one-way hashes, so that individual nodes cannot spoof the base station and thereby flood the network. For unicast packets, nodes must first communicate through the base station, allowing the base station to act as a packet filter to prevent DOS via a single node. INSENS is similarly resilient to distributed DOS or DDOS, because multiple nodes will also not be able to broadcast to the entire network. Second, to prevent advertisement of false routing data, control routing information must be authenticated. A key consequence of this approach is that the base station always receives correct partial knowledge of the topology. Though the base station may not receive all of the topology discovery information, due to localized intrusions, the picture of the network that the base station is able to construct is nevertheless correct. Third, to address resource constraints, INSENS follows two design decisions: symmetric key cryptography is chosen for confidentiality and authentication between the base station and each resource-constrained sensor nodes, since it is considerably less compute-intensive than public key cryptography; and, the base station is chosen as the central point for computation and dissemination of the routing tables. Fourth, to address the notion of compromised nodes, redundant multipath routing is built into INSENS to achieve secure routing. The paths are designed to be disjoint, so that even if an intruder takes down a single node or path, secondary

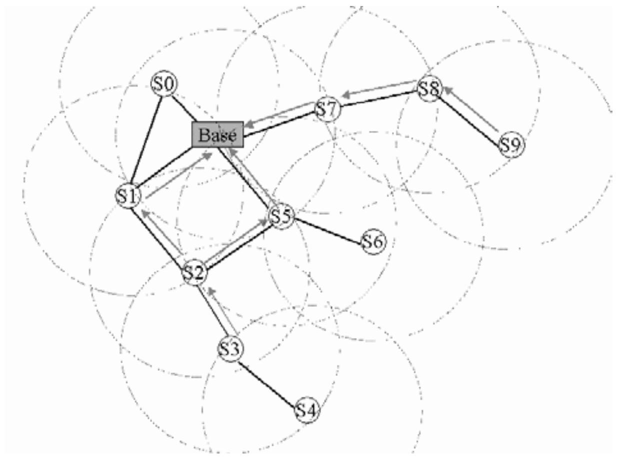


Figure 1. Sample asymmetric WSN topology over 10 sensor nodes with multiple paths to the base station.

paths will exist to forward the packet to the correct destination.

In the remainder of the paper, we address related work in Section 2, provide a detailed description of the INSENS system in Section 3, present implementation and experimental analysis of INSENS in Section 4, and conclude the paper.

2 RELATED WORK

Sensor network security is a critical issue in sensor network research [Perrig00, NAI]. Ganesan et al propose a redundant "multipath" routing approach for a sensor network [Ganesan02] in order to provide fault tolerance and reliable data dissemination. Every node can have multiple paths to another node. Two kinds of multipath are studied: disjointed paths, and braided paths. DPSP [Papadimitratos02a] provides a fast multipath routing algorithm, based on a novel heuristic that picks a set of highly reliable paths. For our approach, multipath routing is useful to combat intrusions or malicious nodes, but is tightly integrated into a complete secure routing system capable of also ensuring authentication and data integrity.

Security and intrusion tolerance approaches based on Byzantine fault tolerance have been proposed [Awerbuch02, Pathak02]. For example, a mechanism for distributed authentication that can tolerate Byzantine faults has been proposed [Pathak02]. It involves a distributed system of mutually authenticating semi-trusted parties. While this mechanism is well-suited for mobile networks, its computational requirements limits its utility for resource-constrained WSNs. Byzantine fault-tolerance algorithms typically require significant computation and communication.

In the field of ad hoc wireless networking [Broch98, Royer99], previous work on secure routing employs public key cryptography to perform authentication ([Kong01, NAI, Papadimitratos02b, Zhou99, Zhang98]). Unfortunately, resource constraints in sensor network limit the applicability of these current public/asymmetric key standards.

SPINS addresses secure communication in resource-constrained sensor networks, introducing two low-level secure building blocks, SNEP and μTESLA [Perrig01]. A brief example of secure basic routing with these building blocks is described. We leverage some of these concepts to implement intrusion-tolerant multi-hop routing for WSNs. For example, we utilize keyed message authentication codes (MAC) similar to SNEP to verify the integrity of control packets. Keyed MAC's are vital for verifying the integrity of topology information delivered to the base station. We also employ the concept of a one-way hash chain seen in μTESLA, but use the chain to provide one-way sequence numbers for loose authentication of the base station, rather than as the key release mechanism seen in SPINS. One-way sequence numbers are essential for limiting a variety of DOS and rushing attacks, as described later.

Instead of a public/private key system, SEADS [HuWMCSA02] and Ariadne [HuMobi02] use symmetric cryptography, a one-way hash function, TESLA [Perrig01], and MACs to build secure wireless network routing. SEADS proposes a secure mechanism on top of DSDV. The paper utilizes secure one-way hash chains to authenticate metric and sequence numbers. The shared secret key between neighbor nodes is used to authenticate neighbors. Ariadne provides a secure routing mechanism built on top of DSR [Johnson96]. Three kinds of mechanisms, e.g. shared secret keys, TESLA, and digital signatures, are proposed for authentication. TESLA is used to authenticate the path between two nodes. Ariadne proposes to use multipath to thwart the effects of routing misbehavior. The multipath is a byproduct of standard DSR. These two protocols propose mechanisms to build routes between two peer nodes in ad hoc wireless networks.

In contrast to a peer-based routing architecture, INSENS constructs network routing for an asymmetric or hierarchical architecture consisting of a base station and sensors. As a result, INSENS's protocol and security architecture are far different. In INSENS, each node shares a secret key only with the base station, and not with any other nodes. This has the advantage in case a node is compromised that an intruder will only have access to one secret key, rather than the secret keys of neighbors and/or other nodes throughout the network. In addition, setting up keys is straightforward in INSENS; each node needs to be programmed with only one secret key for authenticating itself to the basestation, and one initial key for authenticating the basestation to each node. The request-response cycle of INSENS's route discovery phase follows the basic paradigm of the DSR protocol's Route Discovery process. Our contribution is to make such a paradigm secure in a resource-constrained wireless sensor network.

3 PROTOCOL DESCRIPTION

3.1 Design Principles

We have designed and implemented a secure and INtrusion-tolerant routing protocol for wireless SEnsor NetworkS (INSENS). As noted in Section 1, INSENS's design is based on three principles:

- Exploit redundancy to tolerate intrusions without any need for detecting the node(s) where intrusions have occurred. INSENS operates correctly in the presence of (undetected) intruders.
- Perform all heavy-duty computations at the base station(s), and minimize the role of sensor nodes in building routing tables, or dealing with security and intrusion-tolerance issues. INSENS minimizes computation, storage, and bandwidth requirements at the sensor nodes.
- Limit the scope of damage done by (undetected) intruders by limiting flooding and using appropriate authentication mechanisms. INSENS uses symmetric-key cryptography to implement these mechanisms.

The first principle addresses the fundamental problem of the difficulty of detecting intrusions in a timely manner in sensor networks. The values of some of the important parameters, such as normal usage and communication patterns, needed for (anomaly-based) intrusion detection are typically not known in advance in a sensor network, particularly in a critical scenario. Determining these values is time-consuming, and the presence of intruders can make it extremely difficult to determine these values. Thus, anomaly-based intrusion detection techniques cannot be used to detect intrusions. Furthermore, signature-based intrusion detection techniques cannot be fruitfully employed here due to a lack of any experience with most sensor-network based applications and the types of attacks that may be launched.

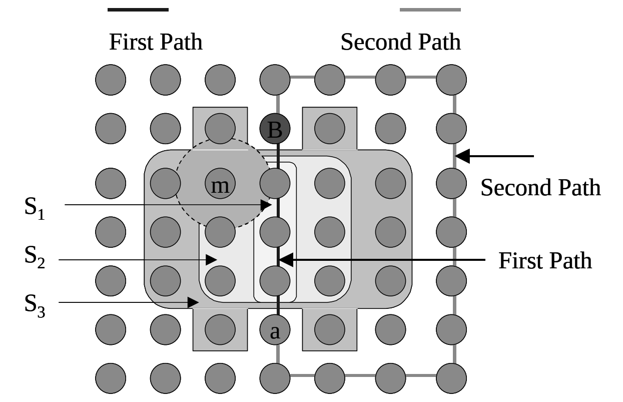


Figure 2: The multipath routing policy selects a second path (red) that shares as few common nodes with the first path (black) as possible.

Rather than rely on traditional intrusion-detection techniques, INSENS's strategy is to design a routing mechanism that is intrusion-*tolerant*. The building or updating of correct routing tables and the correct delivery of messages are performed in a manner that is robust to the presence of a small number of undetected intruders.

INSENS incorporates redundancy in routing to bypass intruders while routing messages. As shown in Figure 2, multiple routes are derived between each source and destination. These paths are independent of one another in the sense that they share as few common nodes/links as possible; ideally, only the source and the destination nodes are shared among paths. Each message sent from a source to a destination is sent multiple times, once along each redundant path. The presence of one or more intruders along some of these paths can jeopardize the delivery of some of the copies of a message. However, as long as there is at least one path that is not affected by an intruder, the destination is will receive at least one copy of the message that has not been tampered with. Notice that this approach works despite the presence of (undetected) intruders.

In Figure 2, the first path between the base station B and the node *a* is chosen as the shortest path. The area shaded gray surrounding the first path consists of all nodes that could be affected by a malicious intruder. For example, a malicious node *m* is shown as the neighbor to a node on the first path. Such a node *m* could disable both a node on the first path as well as its own north-south-east-west neighbors. As a result, the set of nodes (gray area in Figure 2) that should be removed from consideration for the second path consist of all nodes in the first path between the source and destination, all of their neighbors, and all of their neighbors' neighbors. A valid second path is shown avoiding the first path's set of affected nodes.

An important issue here is enabling a destination node to determine which received copies of a message are original, and which have been tampered with. This can be addressed by leveraging appropriate confidentiality, integrity, and authentication mechanisms [Perrig01] while exchanging messages.

The second principle addresses the issue of resource constraints in sensor networks. INSENS takes advantage of the asymmetric topology of the WSNs. Since base stations in sensor networks are resource rich, and sensor nodes are resource constrained, our protocol minimizes the use of important resources such as CPU, memory, bandwidth or power at the sensor nodes at the expense of increased computation, communication, storage and power requirements at the base stations. The overall structure of the protocol for building forwarding table for each node follows three phases. The base station first sends out a request message, then collects topology information from all sensor nodes, and finally computes and downloads the routing tables (including redundant paths) into each node. This reduces the role of the sensor nodes to simply conveying the appropriate (local) topological information to the base station.

The third principle addresses the issue of damages an (undetected) intruder may cause while the routing table is being built. Clearly, if sufficient care is not taken, intruders can provide false connectivity information or advertise incorrect routes that will result in building incorrect routing tables. Also, intruders can launch DOS attacks by repeatedly sending many copies of the same message, or by sending spurious messages. This may delay indefinitely and even prevent the building of routing tables. INSENS employs the one-way authentication mechanism proposed in [Perrig01] to authenticate any information sent by the base station, and appropriate integrity mechanisms to ensure that any tampering with the information being exchanged can be detected by the intended receiver. Tamper detection ensures that the base station is able to glean out the correct (untampered) information from all the messages it receives from sensor nodes. In addition, INSENS limits flooding of messages by allowing communication only between the base station and the sensor nodes, and by having sensor nodes drop duplicate messages. These techniques essentially limit the damage an intruder may cause. Together, these design choices ensure that an intruder may be able to take out a small part of the network, but cannot compromise the entire network.

3.2 Route Discovery

Route discovery ascertains the topology of the sensor network and builds appropriate forwarding tables at various nodes. Route discovery is performed in three rounds. In the first round, the base station floods (limited flooding) a *request message* to all the reachable sensor nodes in the network. In the second round, sensor nodes send their (local) topology information using a *feedback message* to the base station. In the third round, the base station computes the forwarding tables for each sensor node based on the information received in the second round and sends them to the respective nodes using a *routing update* message.

3.2.1 First Round: Route Request

The base station initiates the first round whenever it needs to construct the forwarding tables of all sensor nodes. This can be in the beginning when the network has just been established, or when the network may have changed substantially due to node mobility. The

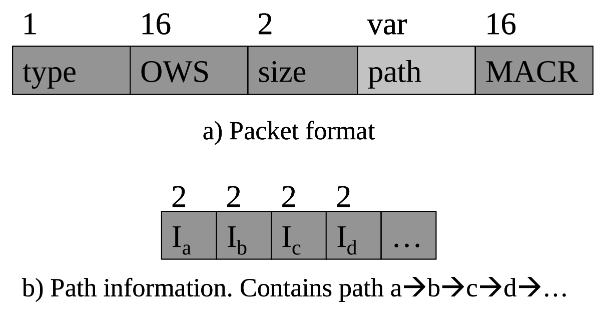


Figure 3. Route *request* format

base station broadcasts a request message that is received by all its neighbors. A sensor node that receives a request message for the first time in turn broadcasts a request message. A request message broadcast by a node *x* includes a path from the base station to *x*. When a node receives a request message for the first time, it forwards (broadcasts) this message after appending its identity in the path. It also records the identity of the sender of this message in its neighbor set. When a node receives duplicate request messages, the identity of the sender is added to its neighbor set, but the duplicate request is not rebroadcast.

Propagation of request messages in this way serves three purposes: (1) it informs all sensor nodes that the base station is collecting topology information to build forwarding tables, (2) it aids in constructing a path from each sensor node to the base station that is used in the second round to forward feedback messages to the base station, and (3) a node receiving a request message learns that the sender of that message is its neighbor.

A malicious node in the network can attempt to launch several attacks in this round. First, it can attempt to spoof the base station by sending a spurious request message. Second, it can include a fake path in the request message it forwards. Third, it may not forward a request message, or launch a DOS attack by repeatedly sending several request messages. We use two mechanisms to counter these attacks. Both of these mechanisms require sensor nodes to be pre-configured with appropriate values.

First, we leverage the concept of one-way sequences proposed by the μTESLA protocol [Perrig01] to identify a request message initiated by the base station and to restrict DOS-style flooding attacks. The base station generates a sequence of numbers $n_1, n_2, n_3, ..., n_{k-1}, n_k$, such that $n_{i+1} = F(n_i)$, where F is a one-way function, $0 < i < k$, and n_1 is chosen randomly. F has the property that it is computationally infeasible to compute n_{k-1} in a limited time by knowing n_k and F. All sensor nodes are pre-configured with function F and value n_k. The base station transmits n_{k-1} (called a *One-Way Sequence (OWS) number*) in the first request message as shown in Figure 3. When the base station needs to construct forwarding tables again, the second request message originated by the base station will be assigned an $OWS_2=n_{k-2}$. The *i*'th request message will be assigned $OWS_i=n_{k-i}$. All nodes forwarding the *i*'th request message repeat OWS_i in the header. A sensor node receiving the *i*'th request message will compute $F^j(OWS_i)$ for j=1,2,…,J, where $F^j(\#)$ = F(F(...F(#)) applied *j* times. A sensor node will have saved the most up to date or freshest OWS_{fresh} that it has seen from the base station. If OWS_{fresh} is within J applications of the function F to OWS_i from the *i*'th request message, then $F^j(OWS_i)= OWS_{fresh}$ for some *j*. This match enables the sensor node to verify that only the base station could have generated this OWS. If there is not a match, then the packet is deemed spurious and is not forwarded. This policy prevents propagation of spurious messages. Also, messages whose OWS is older than OWS_{fresh} are not forwarded. This policy prevents a node from flooding the network with out of date messages. For example, when a sensor node receives the first request message, it will compare $F(OWS_1)$ with $OWS_{fresh} = n_k$. If there is a match, then the node knows that only the base station could have produced this next OWS in the sequence. Otherwise, the message is deemed spurious and is not forwarded.

A malicious node cannot generate the next OWS number in the sequence. This restricts the ability of a malicious node to spoof the base station. An arbitrary sensor node cannot therefore flood a *new* request message. As mentioned earlier, an intruder is also prevented from flooding *old* request messages. However, it remains possible that a malicious node could flood a modified request message using the *current* OWS from a valid request message just sent out by the base station. In such an attack, called a rushing attack [HuTech02], an attacker tries to propagate a spurious message before the base station can propagate its own valid message. Our defensive countermeasures confine such an attack to the local subtree of nodes below the malicious node. In such an attack, the intruder must first wait to hear the current OWS from the base station before launching its own attack. Since duplicate requests (same OWS) are not rebroadcast, nodes in the tree that are closer to the base station than the malicious node will receive the valid request message first. These nodes will drop the intruder's spurious request messages received later. Moreover, an attacker is restricted to sending only one such request message per OWS, since neighboring nodes will forward a request message as defined by its OWS exactly once. A malicious node cannot launch a DOS attack by sending multiple request messages. Of course, the attacker could pack a long fake path into its only spurious request message. Regardless, as shown in Figure 4, the damage of flooding a spurious request message is locally confined to the nodes nearest to and downstream from the intruder. In the figure, it is assumed that the rest of the network hears the valid request message from the basestation first.

The second mechanism that we use to defend against intrusions, in addition to the one-way sequences, is a keyed MAC algorithm. Each sensor node is configured with a separate secret key that is shared only with the base station. Before forwarding a request message, a node *x* generates a 16-byte MACRequest ($MACR_x$) by applying a keyed MAC algorithm. This MAC is applied to the complete path consisting of the current node *x*'s identity appended to the path from the incoming request message. This 16-byte field is compatible with standard 128-bit MAC algorithms and its overhead is incurred only during the route discovery phase. The correctness of INSENS is not dependent on a specific MAC length. Though increased memory will be needed to store multiple 16-byte MAC's, the current ATMEL processors for the Motes support 128 KB of memory, up from the 8 KB that was the design constraint for SPINS [Perrig01]. Also, another design constraint of the Mote architecture is the 30-byte length of each packet. We assume that future sensor architectures

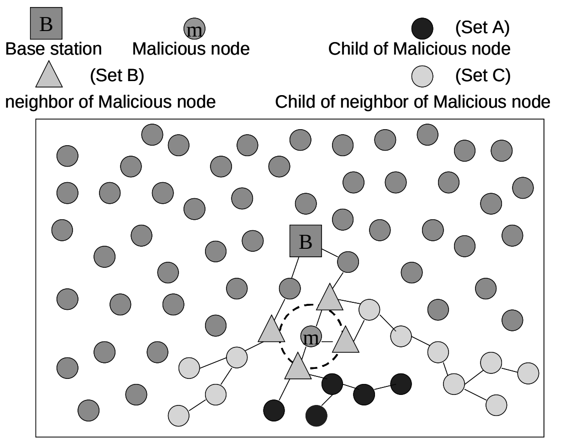


Figure 4. The damage inflicted by a malicious node *m* is confined to a localized portion of the sensor network, i.e. nodes downstream from *m* and downstream from *m*'s neighbors.

will be able to accommodate variable-length packets with the above MAC lengths.

The secret key of the node is used to generate the following MACR:

$$MACR_x = MAC(size \mid path \mid OWS \mid type, Key_x)$$

where "|" denotes concatenation. This MACR is included in the request message as shown in Figure 3(a). It is used to check the integrity of the path in the second round when the nodes receiving the request message need to forward a feedback message to the base station along this path (in the reverse direction). As we shall see, a fake path included by a malicious node cannot be verified in the absence of a correct MAC as the feedback message is forwarded towards the base station, and as a result, the spurious feedback message will be dropped.

The overall effect of these security mechanisms is that a malicious node can attack in the first round only by localized flooding, by not forwarding a request message, and by sending fake path in the request which is later on detected in the second round. The latter two attacks will result in some of the nodes downstream from the malicious node not getting a request message or not being able to forward their feedback message to the base station in the second round. Again, a malicious node may be able to compromise a small number of nodes in its vicinity by employing these types of attacks, but cannot jeopardize the security of the complete network.

Figure 3(a) details the format of a *request message* along with the size (in number of bytes) of each field in the format. The *type* field indicates whether the message is a request, feedback, routing, or data message. The *OWS* field contains the one-way sequence number. The *path* field contains the path (sequence of node identities as shown in Figure 3(b)) from the base station to the current node (the node that sends this request message). The *size* field contains the length of this path. The *MACR* field contains a MAC (message authentication code) of size, path, OWS and type as described above.

3.2.2 Second Round: Route Feedback

In the second round, each sensor node sends its local connectivity information (a set of identities of its neighbor nodes as well as the path to itself from the base station) back to the base station using a *feedback* message. After a node has forwarded its request message in round one, a node will wait a certain timeout interval before generating a feedback message. This interval allows a node to listen to the local broadcasts of its neighbors, who will also be forwarding the same request message. A node will hear the request messages from its upstream, peer and downstream neighbors.

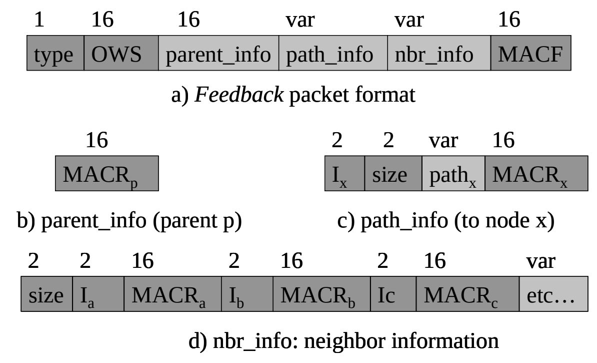

1 16 16 var var 16
type OWS parent_info path_info nbr_info MACF
a) *Feedback* packet format
16
MACR$_p$
b) parent_info (parent p)
2 2 var 16
I$_x$ size path$_x$ MACR$_x$
c) path_info (to node x)
2 2 16 2 16 2 16 var
size I$_a$ MACR$_a$ I$_b$ MACR$_b$ Ic MACR$_c$ etc…
d) nbr_info: neighbor information

Figure 5. Route *feedback* message from node x.

As shown in Figure 5a), a feedback message generated by a node contains its neighborhood information (a set of identities I_i of all its i neighbors, denoted by *nbr_info*), as well as the path to that node from the base station (*path_info*). The path listed in *path_info* is the path through the upstream neighbor who first broadcast this particular request message to the node. This upstream neighbor is denoted as parent *p* in the ensuing discussion. For example, if node *x* receives the first request message for the current OWS from neighbor *c*, then neighbor *c* becomes the parent of neighbor *x*, namely $p_x=c$. If this first request message from *c* contained the path base→a→b→c, then the path returned in node *x*'s *path_info* will be base→a→b→c→x.

The integrity of the topology data returned to the base station by each node in its feedback message must be protected, so that the base station is able to correctly reconstruct the topology of the network. Accordingly, both the list of neighbors *nbr_info* as well as the path *path_info* to node *x* are protected by the following keyed MACFeedback$_x$:

$$MACF_x = MAC(path\_info \mid nbr\_info \mid OWS \mid type, Key_x)$$

The MACF ensures that the base station will construct a correct topology, though it may be incomplete due to malicious nodes that may drop or tamper with feedback messages. The messages that reach the base station are guaranteed after verification to be correct and secure from tampering. It is still possible that a compromised node originates a message that passes tamper inspection, but still provides false neighbor information, i.e. omits some neighbors. This inconsistency will be detected by the base station after round two and prior to round three when the compromised node's neighbor list is compared with the neighbors listed in feedback messages from all the neighbors of the compromised node. It is at this point that the MACR's of each neighbor contained in *nbr_info* (Figure 5d) are used for consistency checks.

Our remaining task is to route the feedback message from node *x* back to the base station. In the absence of malicious nodes, it is straightforward for the feedback message to follow the reverse path taken by the request message that initiated the feedback response. As mentioned earlier, each child node will have already identified its parent as the first of its upstream neighbors to send the child a request message with the current OWS. The linked chain of child-parent pairs creates a reverse unicast path from node *x* back to the base station (multiple paths are not available until after round three). Though it may be more reliable to flood the feedback message back to the base station given intruders, our design has deliberately avoided giving sensor nodes the capability of initiating a flooded message. This is done to prevent global DOS attacks. Sensor nodes are only allowed to unicast and controlled multicast back to the base station. As a result, route feedback messages are subject to attacks that drop unicast packets, though the damage is again confined locally, similar to Figure 4.

Before forwarding a feedback message, a child node should place parent identification information into the *parent_info* field of the feedback message. This *parent_info* determines which of a child's upstream neighbors is the parent who should forward the feedback message. However, simply using the parent address I_p doesn't require the casual attacker to have any knowledge of the local topology nor of the current state of the topology discovery process. Instead, INSENS requires a child node to place its parent's $MACR_p$ into the *parent_info* field. A child node will already have on hand the $MACR_p$ of its parent *p* from the parent's original request message, i.e. from the first request message received by the child. This $MACR_p$ is tightly linked with the current state of the OWS request-feedback cycle, and also to the path to the child node. Thus, the $MACR_p$ serves a security function, in addition to an addressing function. A casual attacker that only knows node ID's would be unable to forward a spurious feedback message because it won't be able to provide a valid address of any of the upstream nodes. A more adept intruder would have to know the up to date $MACR_p$ corresponding to the current OWS in order to launch an attack and have its spurious feedback message be accepted by an upstream node.

The $MACR_p$'s addressing function selects the specific parent from all upstream nodes to forward this feedback message. When an upstream node hears the local broadcast of a feedback message whose $MACR_p$ does not match its own MACR that it originally sent with this OWS, then that upstream node knows that it is not the parent, and should not forward this feedback packet. If the two MACR's match, then the upstream node knows that it has been selected as the child's parent and should forward the feedback packet. In the absence of intruders, only the nodes listed in node *x*'s *path_info* will engage in forwarding the feedback message along the reverse path back to the base station.

The basic mechanism presented thus far for forwarding of the feedback packet is relatively lightweight in terms of computation at each node. The only new computation that must be performed is generation of the MACF by the originating node. Intermediate nodes may have to apply the one-way function F to an OWS that they don't recognize to determine whether it is valid. Otherwise, intermediate nodes don't have to recalculate a MAC and simply engage in logic comparisons of MACR's as well as memory copies of the new $MACR_p$ into the *parent_info* field. This is the only field of the feedback packet that is modified in transit. The *path_info* and *nbr info* in the feedback message aren't changed as the message is propagated back.

Having established the basic format of the feedback message as well as the basic structure for forwarding of the feedback message, we observe that a malicious intruder could still launch several attacks. First, an intruder could launch a DOS-style attack and send multiple feedback messages to each of its upstream neighbors. Second, an intruder could eavesdrop and learn topology information, i.e. the identities and MACR's of neighbors to a remote node x as well as the path to node x. Third, an intruder could divert a feedback message to the wrong upstream node.

To address the first DOS-style attack, we employ two defense mechanisms. First, to prevent repetitive transmissions of a feedback packet from the same originating node, all nodes follow the policy of not forwarding duplicate feedback messages. When an intermediate node receives a feedback message originating from a node x for the current OWS, it checks to see if it has already seen such a message from node x based only on the Type, OWS and Identity (from *path_info*) fields. If such a message has already been seen, then the message is not forwarded. This limits a malicious node to sending only one feedback message per originating node per neighborhood. Once a child node, malicious or not, has sent its first feedback message labeled as originating from node x, then all of the child node's upstream neighbors will remember that they have seen such a message and will not forward any subsequent feedback messages with the same originating node and same valid OWS.

Two types of attacks can be launched against our policy of suppressing duplicate feedback messages: memory exhaustion attacks and rushing attacks. To combat memory exhaustion attacks, we store only 1 bit per node to flag whether an originating ID has been seen before; given a 16-bit address space, this will consume 8 KB per OWS, and the node may choose to save a history of the most recent *N* OWS numbers; *N*=3 will consume 24 KB. If memory trends continue, e.g. Motes currently support 128 KB for code (not data), a malicious node will not be able to overflow this fixed and compact memory allocation. If needed, some other clever strategies can further reduce this memory requirement, e.g. node addresses can be hashed to a much smaller-sized hash table. This hashing strategy may miss catching some feedback messages that hash to the same value. INSENS's defense against rushing attacks in the feedback phase is somewhat limited. In contrast to the request phase where a rushing attack must wait until an up to date OWS has been received, an intruder need not wait for the corresponding feedback message. After receiving a valid OWS in a request, an attack can be launched immediately on the upstream nodes by sending false feedback messages, in advance of any valid feedback messages, thereby causing valid feedback packets to be dropped as duplicates. Such an attack would however terminate at the base station, and would be confined to a localized portion of the network.

The second defense mechanism against DOS-style attacks is to employ rate control to prevent transmissions of feedback packets from many thousands of phantom originating nodes. Note that unlike the propagation of request messages in the first round, a node may in fact forward many feedback messages in the second round. A malicious child node can exploit this to launch a DOS attack by repeatedly sending spurious feedback messages that contain a valid OWS number, any originating ID, and a valid MACR$_p$ of any upstream node. An upstream node (except the base station) has no way of distinguishing between an authentic feedback message and a spurious feedback message generated in the above-mentioned way. A DOS attack in this way will congest the path from the malicious node to the base station. INSENS imposes a rate control mechanism that restricts the rate at which a node may send messages. If a malicious node attempts to send messages at a very fast rate, the upstream (correct) node will forward those messages only at a slower (legitimate) rate. This will prevent congestion on all the further upstream nodes. For example, if the maximum sending rate allowed for a node is 1 kb/s, a correct node will only send at 1 kb/s, irrespective of the rate at which it receives messages from its downstream nodes. Even if a downstream node spoofs 1000 nodes, each sending at 1 kb/s, the upstream node will only send at 1 kb/s, rather than 1000 kb/s.

To provide confidentiality against eavesdropping by a malicious node, the *path_info* and *nbr_info* is encrypted using the originating node x's secret key, with the caveat that the identity field of the originating node in *path_info* is left unencrypted. Thus, for each feedback packet, only the Type, OWS, *parent_info* MACR$_p$, and identity of the originating node or sender are in the clear. No topology information in terms of path or neighbor information is revealed to any intermediate node. The identity of the originating node must be in plain sight so that 1) the base station can determine to whom the topology information in the feedback packet belongs, and 2) duplicate feedback packets can be spotted.

INSENS's defense against diversion of a feedback message to a "wrong" upstream node, i.e. an upstream node that is not listed in the path contained in *path_info*, is based on the principle that it doesn't matter by which path the topology information reaches the base station. In our current approach, an attacker could substitute the MACR of any of its non-parental upstream nodes. This would divert the feedback packet away from its one valid parent. Even if a malicious node diverts a feedback packet off of the reverse path taken by the request packet, the feedback packet will simply follow another path of linked child-parent pairs that lead back to the base station. Also, because of the broadcast nature of wireless medium, the other upstream nodes will hear this feedback message and will suppress any duplicates. This prevents a malicious child node from sending spurious feedback messages with originating node x to each of its upstream neighbors.

The overall effect of these security mechanisms is that a malicious node is limited in the damage it can inflict, whether attacking by DOS attack, by not forwarding a feedback messages or by modifying the neighborhood information of nodes, which can be detected at the base station. The rate-controlled DOS attack will affect upstream nodes, but only in a limited way. The latter two attacks will result in some of the nodes downstream from the malicious node not being able to provide their correct connectivity information to the base station. Though a malicious node could launch a battery-drain attack by persistently sending spurious feedback messages at the rate-controlled limit, such an attack would still affected a limited number of upstream nodes. In summary, a malicious node may be able compromise a small number of nodes in its vicinity using these attacks, as in Figure 4.

3.2.3 Third Round: Routing Table Propagation

After sending the request message in the first round, the base station waits for a certain period of time to collect all the connectivity information received via feedback messages. A very important consequence of the security mechanisms used in the first two rounds is that the base station can glean out all the connectivity information that has not been tampered with. After receiving a feedback message, the base station recomputes MACFx and verifies that there is a match. If there is a match, then the base station attempts to match the nodes listed as neighbors with prior information received by the base station. The MACR's in *nbr_info* received from neighbors should be consistent with the MACR's reported back to the base station. The MACR is proof that the neighbors heard each others' individualized rebroadcasts of the request message, and that phantom node identities were not simply listed as neighbors. For example, suppose that node *y* first reports back that node *x* is its neighbor, and provides the MACRx contained in *x*'s rebroadcast of the current request message. Later, node *x's* feedback message arrives at the base station. At this point, to verify that node *y* is the neighbor of *x*, the base station matches the MACRx reported by x with the MACRx reported observed by y. If the two match, then there is consistency. Further, the base station matches the MACRy reported by y with the MACRy observed by x. If the two match, then there is complete agreement that the two are neighbors in the topology.

1	16	2	var	16
type	OWS	size	Forwarding table	MAC

Figure 6. Routing table update message.

From this connectivity information, the base station then computes the forwarding tables of each node in the network. In addition to being able to authenticate the connectivity information, there are several other advantages of this strategy of base station computing all the forwarding tables. First, since computing forwarding table involves additional computations, this strategy reduces computation at the sensor nodes. Second, since the base station has the complete information about the network, it can do a better job in selecting appropriate routes in terms of balancing the routing load on the sensor nodes and using appropriate algorithms to select redundant routes that minimize the extent of damage a malicious node may cause.

An important goal of INSENS is to minimize the damage a malicious node may inflict. In particular, a malicious node has a greater chance of inflicting damage on nearby nodes, for example by launching a DOS attack. So, INSENS attempts to choose two independent paths in such a way that the nodes in the two paths are far apart. The first path is chosen using Dijkstra's shortest path algorithm as described above. The second path is computed as follows. Referencing

Figure 2, remove the set S_1 of all nodes that belong to the first path from the connectivity information. Next, remove the set S_2 of all nodes that are neighbors of the nodes in S_1. Next, remove the set S_3 of all nodes that are neighbors of the nodes in S_2. Now compute a shortest path from this updated network connectivity information. If a path is found, it will be the second path. If no path is found in this step, put back the nodes of set S_3 and the corresponding edges in the network connectivity information, and compute a shortest path. If a path is found, it will be the second path. Again, if no path is found in this step, put back the nodes of set S_2 and the corresponding edges in the network connectivity information, and compute a shortest path. If a path is found, it will be the second path. Depending on the network topology, it is possible that no second path is found. In that case, the current implementation of INSENS maintains only a single path. Notice that there are other interesting strategies possible for finding multiple redundant paths that may not include a shortest path. One future direction of our research is to investigate these strategies.

After computing the redundant paths for each node, the base station computes the forwarding tables of each node. These forwarding tables are propagated to the respective nodes in an breadth-first manner. The base station first sends the forwarding tables of all nodes that are its immediate neighbors. It then sends the forwarding tables of nodes that are at a distance of two hops from it, and so on. This mechanism cleverly uses the redundant routing mechanism just built to distribute the forwarding tables. Standard security techniques such as those proposed in [Perrig01] can be used to distribute these forwarding tables in a secure manner.

The structure of these forwarding tables is described in the next subsection. Figure 6 shows the format of the *routing table message* used to propagate the forwarding tables. Fields *Type* and *OWS* are the same as those in the corresponding request or feedback messages. *Dest* contains the address ID of the destination node *x*. *Size* contains the length of the message, and *forwarding table* contains the forwarding table for node *x*. The *forwarding table* entry is encrypted using the secret key of *x*. The *MAC* contains the MAC of the complete message generated using the secret key of *x*. If a routing table message is too long, the base station can segment it and send each segment separately.

3.3 Forwarding Data

Using the forwarding tables built in the route discovery phase, data is forwarded from source (sensor) nodes to the base station, and from base station to the destination (sensor) node. A node maintains a forwarding table that has several entries, one for each route to which the node belongs. Each entry is a 3-tuple: destination, source, and immediate sender. Destination is the node id of the destination node to which a data packet is sent, source is the node id of the node that created this data packet, and immediate sender is the node id of the node that just forwarded this packet. For example, given a route from node S to D: S→a→b→c→D, the forwarding table of node a will contain an entry <D, S, S>, forwarding table of b will contain an entry <D, S, a>, and the forwarding table of c will contain an entry <D, S, b>. The reason for including the node id of the immediate sender in a forwarding table entry is that a node may receive a packet with the same source and destination node many times, because each packet is forwarded over multiple routes. For example, if the other route from S to D is S→e→f→g→h→D, and b and h are neighbors, b will receive the data packet forwarded by h, which it should not forward. This is accomplished by including the immediate sender field.

With forwarding tables constructed in this way, forwarding data packets is quite simple. On receiving a data packet, a node searches for a matching entry (destination, source, immediate sender) in its forwarding table. If it finds a match, it forwards (broadcasts) the data packet.

4 IMPLEMENTATION AND PERFORMANCE

We have simulated INSENS on ns2click, a network simulation tool that combines the ns-2 network simulator [4] with the Click Modular Router[5]. Ns2click was developed in the Computer Science

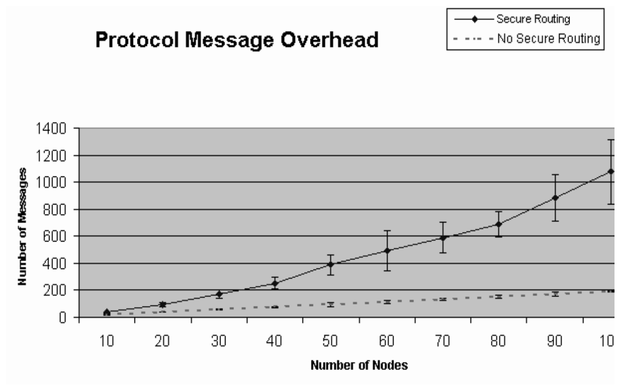


Figure 7. Routing overhead of secure multipath three-phase protocol versus a highly optimistic insecure single-path one-phase protocol.

Department at the University of Colorado, Boulder, and has been used to experiment with several routing protocols in wireless and mobile networks. While ns-2 is a popular network simulator that has been used by many researchers, click is a relatively new routing emulator developed at MIT. Ns2click provides an easy way to develop, debug, experiment with, and deploy network protocols, since the same Click code can be run on an actual system as well as under the simulator ns2.

In our simulation, we implemented our own Click element to simulate the behavior of INSENS on sensor nodes and the base station. Ns-2 was used to simulate the wireless network environment, including the MAC (Medium Access Control) protocol and the lower layers of the wireless network, as well as the geographic distribution of nodes.

Based on this simulation, we have analyzed four aspects of INSENS: (1) overhead of the protocol in terms of number of packets exchanged, packet size, and total number of bytes transmitted, (2) its ability to withstand malicious attacks during route discovery, (3) withstanding malicious attacks during data forwarding, (4) performance against DOS attacks.

4.1 Protocol Overhead

We have performed two sets of experiments to measure the overhead of INSENS. The first set of experiments measure the total number of packets (as part of request, feedback, and routing update messages) exchanged during route discovery. We have measured this for many sensor networks

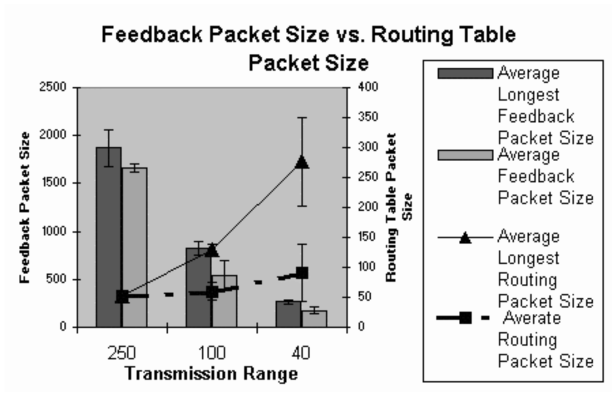


Figure 8. As transmission range decreases, the network becomes less dense and more multi-hop, increasing the routing table size but decreasing the size of the feedback packet. Deviation bars are shown.

consisting of different numbers of nodes. Figure 7 plots the average number of packets exchanged during route discovery as a function of the number of sensor nodes in the network. For a given number of nodes, we generated 20 different network topologies at random. All these topologies maintain the same node density (100 nodes per 1700 * 1700 m^2). The transmission range was set to 250 m. The average number shown by dots in Figure 7 is the average number of packets exchanged over these 20 different network topologies, and the vertical lines along these dots show the variance.

To provide a benchmark for comparison, we designed a trivial routing protocol that has no security or intrusion tolerance. In this protocol, the base station sends a request message that is forwarded by all sensor nodes exactly once. In addition, each node records the identity of the node from which it receives the request message first time as its parent node. After an interval, nodes begin to send “feedback” message to its parent node. As the feedback message is propagated towards the base station, each sensor node updates its forwarding table. A node forwards a feedback message to its parent after appending its children information. Eventually, the base station receives feedback messages from all its neighbors and computes the complete topology of the network. The total number of packets exchanged in this trivial protocol is 2N, where N is the number of sensor nodes.

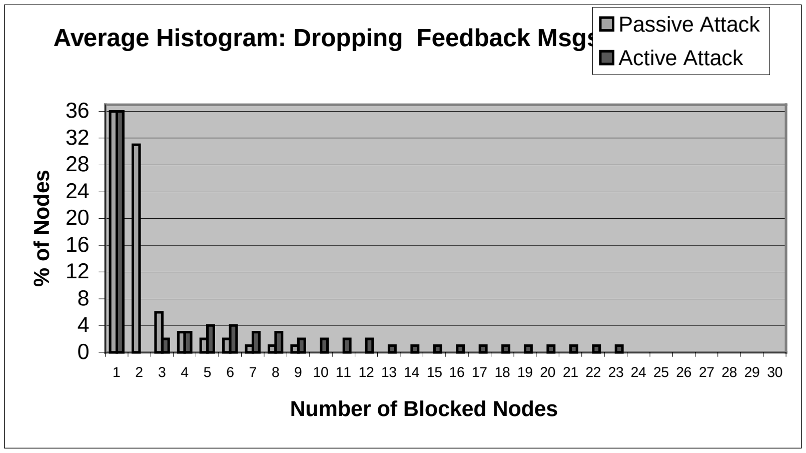


Figure 9. For a given topology of 100 nodes, each node becomes the intruder dropping the feedback messages, and the # of blocked nodes is counted. This creates a histogram. We generate 20 such topologies and plot the averaged histogram.

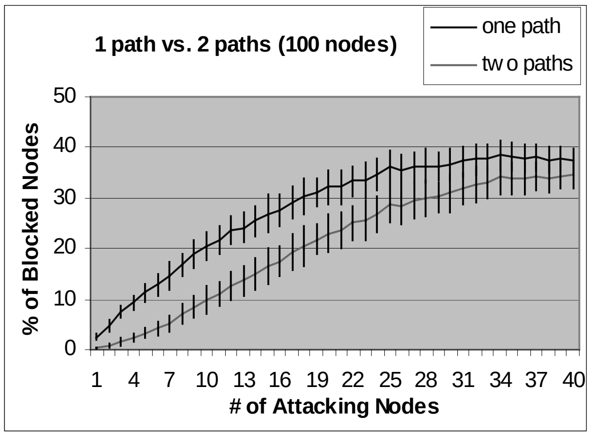
We compare INSENS with this trivial protocol in Figure 7. It is clear that INSENS sends more packets than the trivial protocol, and the difference increases with increasing numbers of nodes in the network. This difference is attributed to the overhead involved in dealing with security and intrusion-tolerance issues.

In the second set of experiments, we measure the effect of node connectivity on protocol overhead. To do this, we deploy 100 nodes in a $300 \times 300m^2$ area and generate a node layout at random. We change the connectivity of the nodes by changing the transmission range from 250m to 40m. Note that a larger transmission range implies higher connectivity. We have measured the size of feedback and routing update packets in these experiments. Figure 8 plots these packet sizes averaged over 20 different topologies as a function of transmission range. For both types of messages, we report the average packet size, and average of the largest packets over the 20 topologies. This figure shows that as the node connectivity increases, the feedback packet size increases and the routing update packet size decreases. The reason for increase in feedback packet size is that with increase in node connectivity, the size of neighbor-union increases; each node has more number of neighbors. The reason for decrease in the routing update packet size is that with increase in node connectivity, the distance between a node and the base station becomes smaller. As a result, the number routes to which a node belongs also becomes smaller.

4.2 Malicious Attacks during Route Discovery

As mentioned in Section 3, a malicious node may be able to compromise a small set of nodes in its vicinity during route discovery. We performed a set of experiments to measure the extent of damage a malicious node can cause during route discovery. We have simulated two types of attacks a malicious node may launch. In the *passive attack*, a malicious node either drops feedback messages or modifies the neighbor information in the feedback message before forwarding (recall that this tampering is later on detected by the base station). The effect of passive attack is that some of the nodes may not be able to convey their connectivity information to the base station and hence will not be included in the network topology constructed by the base station.

In the *active attack*, a malicious node launches a man-in-the-Middle attack. Using this attack, a malicious node may lead two of its neighbors to believe that there is a direct link between them. This attack is a special case of a wormhole attack, in which an attacker creates a tunnel between two nodes to mislead neighbors and/or inject false packets. Note that a man-in-the-middle attack can be avoided by a clever mac-layer design. Figure 9 reports the maximum damage a malicious node may cause by



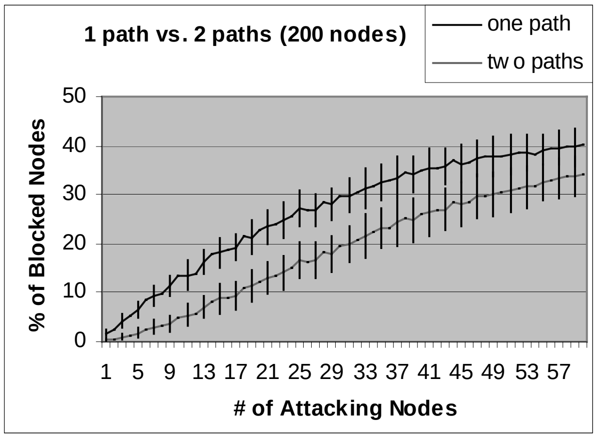


Figure 10. Multi-node attack on a sensor network that has secure single path and multipath routing. Left graph shows 100 nodes, and right graph shows 200 nodes. X axis: # of attacking nodes. Y axis: # blocked nodes unable to send packets. This passive attack corresponds with tolerating faulty nodes.

launching active and passive attacks. The x-axis in this graph records the maximum number of nodes that may be compromised by a single malicious node, and the y-axis records the number of such (malicious) nodes. For example, the red/dark bar on x-axis with value=6 shows that there are four nodes in the network, such that if any of these four nodes turns malicious and launches an active attack, it can compromise a maximum of 6 nodes in its vicinity.

The numbers reported in this figure are averaged over 100 different randomly generated topologies of 100 nodes distributed over a $2000 \times 2000 m^2$ space. In case of active attack, we have calculated this damage by counting all the nodes downstream from the malicious node, its neighbors, and the neighbors' downstream nodes. In case of passive attack, we have calculated this damage by counting all the nodes downstream from the malicious node.

From this figure, we can see that an active attack compromises more nodes than the passive attack, and a significant majority of the nodes can compromise only a small number of nodes. For example, 75% of the nodes can only compromise less than 4 nodes using a passive attack.

4.3 Malicious Attacks during Data Forwarding

INSENS builds multiple paths to bypass malicious nodes. With two independent routes available between every node and the base station, our protocol's goal is to route messages correctly in the presence of a single malicious node. Interestingly, our protocol deals quite well with multiple malicious nodes as well. We have performed a set of experiments to measure the number of nodes that can be blocked when a set of multiple nodes turn malicious and drop data packets. Figure 10 shows the average number of nodes that can be blocked as a function of the number of malicious nodes. For comparison, we have also calculated this number when a single-path routing algorithm is used instead.

These results are based on a network of 100 nodes and 200 nodes randomly distributed over a $1500 \times 1500 m^2$ space. The numbers reported in this figure are averaged over 50 different combinations of nodes randomly selected to be malicious. For example, for 10 malicious nodes, we measured the number of blocked nodes for 50 different combinations selected randomly of 10 nodes turning malicious. For each test, 20 random topologies were chosen.

This figure shows that INSENS does reduce the number of nodes that can be blocked over a single-path routing protocol. Also, even when a relatively large number of nodes are malicious, only a relatively small number of nodes are blocked. For example, in both tests, when 10% of nodes turn malicious, they

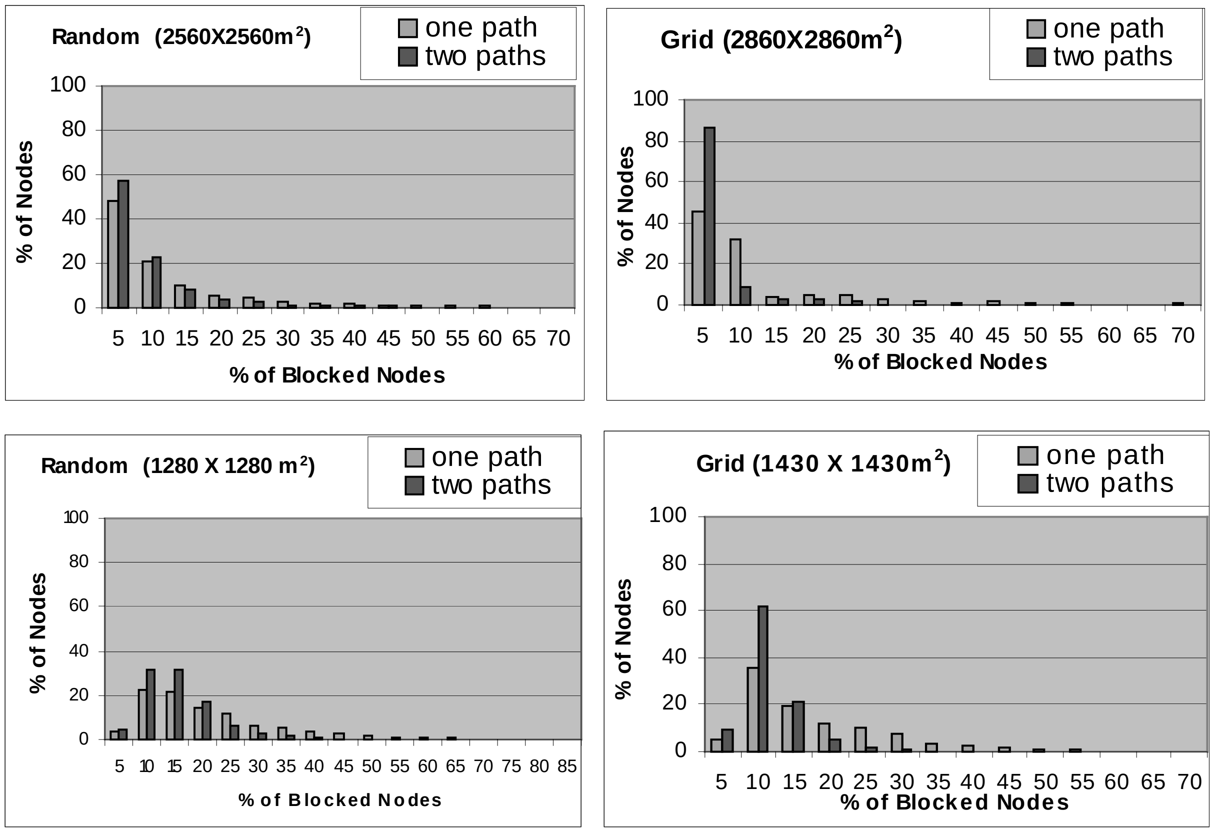


Figure 11. The histogram of DOS simulation with single and multiple paths. Each node becomes the DOS intruder, and the percentage of other blocked nodes is counted given secure single-path and multipath routing. This generates a histogram per topology. For both random and grid topologies, we generate 50 such topologies and plot the averaged histogram.

can block only another 10% or less number of nodes in the network.

4.4 DOS Attacks

Finally, we have performed a set of experiments to analyze the effect of DOS attacks that a malicious node may launch. The DOS attack we have simulated in these experiments is comprised of repeatedly sending data packets to the base station to block the wireless medium and not allow other nodes to send their data packets. DOS attacks are difficult to address completely at the network level. In our opinion, these attacks must be addressed at multiple levels. In our analysis, we have assumed the following: (1) Sensor nodes use an appropriate rate-based control mechanism while forwarding data packets. This implies that a malicious node that repeatedly sends data packets will be able to block its neighbors, but not other (upstream) nodes. (2) The base station has sufficiently large bandwidth available so that a malicious sensor node in its vicinity cannot block the base station by using a DOS attack.

Figure 11 shows the damage a malicious node may cause by launching a DOS attack. The damage caused by a DOS attack depends on the effectiveness of multipath routing, the density of interconnection of the sensor network, and the topology of the graph. In this experiment, two network densities (sparse and dense) and two topologies (random and grid) are tested. In random generated topologies, the position of each node is randomly selected, and the base station is positioned in the center. The total number of nodes is 200. In the grid topology, each node is placed on a square grid. To accommodate the simulator, it was necessary to perturb each position to a small region around each vertex in a square grid graph. In this way, random topologies could be generated even for a nearly uniform square grid. The total number of nodes is 195.

Figure 11 reveals the performance of INSENS against this type of DOS attack. The x-axis records the percentage of nodes that may be blocked by DOS attack launched by a single malicious node, and y-axis records the percentage of such (malicious) nodes. From this figure, we can see that the protection against DOS attacks varies significantly across different network densities and different topologies. As expected, in all cases, the multipath algorithm provides better protection against DOS attacks than the single path approach. The multipath approach performs far better than single path for the grid topology, because the grid nearly always offers a valid redundant second path. The best performance of the multipath approach is obtained for sparse grids (upper right graph), where 85% of intruder nodes are limited to blocking five or fewer nodes. The sparseness limits an intruder to blocking only a few nodes, while the grid almost always offers the sender a valid secondary path. The worst performance of the multipath approach is obtained for sparse random topologies (upper left graph), in which nodes have few neighbors and there are few alternative paths (usually only one path) to the base station. This prohibitively limits the multipath approach, which performs only slightly better than single path routing in this case.

As the network becomes denser, moving from the top row of graphs to the bottom row in Figure 11, attackers are able to block increasing numbers of nodes, and the histograms shift to the right. This is true for both random and grid topologies.

5 CONCLUSIONS

In this paper, we have developed INSENS, a secure and INtrusion-tolerant routing protocol for wireless SEnsor NetworkS. Redundant multipath routing improves intrusion tolerance by bypassing malicious nodes. INSENS operates correctly in the presence of (undetected) intruders. To address resource constraints, computation on the sensor nodes is offloaded to resource-rich base stations, e.g. computing routing tables, while low-complexity security methods are applied, e.g. symmetric key cryptography and one-way hash functions. The scope of damage inflicted by (undetected) intruders is further limited by restricting flooding to the base station and by having the base station order its packets using one-way sequence numbers. An important property of INSENS is that while a malicious node may be able to compromise a small number of nodes in its vicinity, it cannot cause wide-spread damage in the network. Performance measured from a prototype implementation using the ns2click simulation tool shows that INSENS tolerates malicious attacks launched by intruder nodes, and functions correctly over a variety of sparse, dense, random and grid topologies despite intrusions.

6 REFERENCES

[**Awerbuch02**] B. Awerbuch, D. Holmer, C. Nita-Rotaru and H. Rubens, "An On-Demand Secure Routing Protocol Resilent to Byzantine Failures," ACM Workshop on Wireless Security (WISE) 2002, pp. 21-30.

[**Bellare96**] M. Bellare, R. Canetti and H. Krawczyk, "Keying hash functions for message authentication, "Advances in Cryptology -- CRYPTO '96, Lecture Notes in Computer Science, vol. 1109, Springer-Verlag, 1996, pp. 1--15.

[**Broch98**] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, ``A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols." Proc. of MobiCom '98, Oct. 1998, pp. 85-97.

[**Click**] Click Web Site, http://www.pdos.lcs.mit.edu/click/.

[**Ganesan02**] D. Ganesan, R. Govindan, S. Shenker and D. Estrin, "Highly Resilient, Energy Efficient Multipath Routing in Wireless Sensor Networks," Mobile Computing and Communication Review (MC2R) Vol 1., No.2. 2002.

[**HuMobi02**] Y. Hu, A. Perrig, D. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom 2002).

[**HuTech02**] Y. Hu, A. Perrig, D. Johnson, "Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols," Technical Report TR01-384, Department of Computer Science, Rice University, June 2002.

[**HuWMCSA02**] Y. Hu, D. Johnson, A. Perrig, "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks," Fourth IEEE Workshop on

Mobile Computing Systems and Applications (WMCSA '02).

[**Johnson96**] D. Johnson and D. Maltz. "Dynamic Source Routing in Ad Hoc Wireless Networks," In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, Chapter 5, pages 153-181, Kluwer Academic Publishers, 1996.

[**Kong01**] J Jiejun Kong, P. Zerfos, H. Luo, S. Lu, Lixia Zhang, "Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks," International Conference on Network Protocols (ICNP 2001).

[**NAI**] NAI Lab, http://www.nai.com/nai_labs/asp_set/

crypto/crypt_senseit.asp.

[**NS**] NS2 Web Site, http://www.isi.edu/nsnam/ns/.

[**ns2click**] NS2Click software simulator Web Site, http://systems.cs.colorado.edu/Networking/ns2click.html.

[**Papadimitratos02a**] P. Papadimitratos, Z. Haas, E. Sirer, "Path Set Selection in Mobile Ad Hoc Networks," The 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002).

[**Papadimitratos02b**] P. Papadimitratos, Z. Haas, "Secure Routing for Mobile Ad hoc Networks," Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002).

[**Pathak02**] Pathak, Iftode, "Byzantine Fault Tolerant Authentication," Poster at ACM WISE Workshop, 2002.

[**Perrig00**] A. Perrig, R. Szewczyk, V. Wen, A. Woo, "Security for SmartDust Sensor Network," http://www.cs.berkeley.edu/~vwen/classes/f2000/cs261/project/sensor_security.html .

[**Perrig01**] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, "SPINS: Security Protocols for Sensor Networks," Proceedings of Seventh Annual International Conference on Mobile Computing and Networks MOBICOM 2001, July 2001.

[**Rivest92**] R.Rivest. "The MD5 Message-Digest Algorithm." Request For Comments: 1321, April 1992.

[**Royer99**] E. Royer, C. Toh, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks," IEEE Personal Communications, vol. 6 no. 2, April 1999.

[**Zhang98**] K. Zhang. "*Efficient protocols for signing routing messages*." In Proceedings of the Symposium on Network and Distributed Systems Security (NDSS '98), San Diego, California, March 1998.

[**Zhou99**] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," IEEE Network Magazine, vol. 13, no.6, November/December 1999.