

# A Performance Evaluation of Intrusion-Tolerant Routing in Wireless Sensor Networks

Jing Deng, Richard Han, and Shivakant Mishra

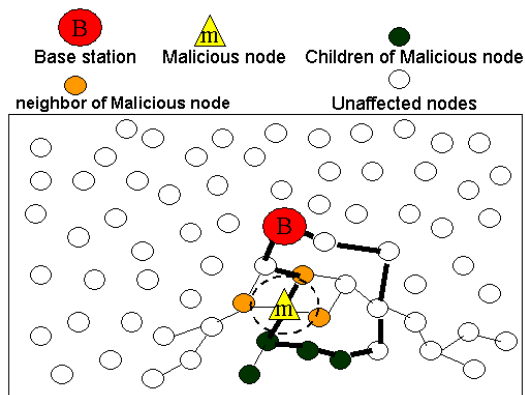
University of Colorado at Boulder, Computer Science Department  
{jing,rhan,mishras}@cs.colorado.edu

**Abstract.** This paper evaluates the performance of INSENS, an INtrusion-tolerant routing protocol for wireless SEnsor Networks. Security in sensor networks is important in battlefield monitoring and home security applications to prevent intruders from eavesdropping, from tampering with sensor data, and from launching denial-of-service (DOS) attacks against the entire network. The resilience of INSENS's multipath performance against various forms of communication-based attacks by intruders is evaluated in simulation. Within the context of INSENS, the paper evaluates implementations on the motes of the RC5 and AES encryption standards, an RC5-based scheme to generate message authentication codes (MACs), and an RC5-based generation of one-way sequence numbers.

## 1 Introduction

Wireless sensor networks (WSNs) are rapidly emerging as an important new area in the research community. 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 settings. For military settings, 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 [1] [2] [3]. To address these issues, this paper develops a secure routing system that is resilient to attempts to obstruct data delivery, and in so doing also develops end-to-end data integrity checksums and authentication schemes that can be used to detect tampering with sensor data.

The design and implementation of secure routing in WSNs must simultaneously address three 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.



**Fig. 1.** Sample asymmetric WSN topology rooted at the base station. Triangle node is a malicious node. Black nodes are its downstream nodes. Intrusion-tolerant routing is assisted by multiple paths; downstream nodes can still communicate with the base station.

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. Third, WSNs face the added physical security risk of being deployed in the field, so that individual sensor nodes can be obtained and subject to attacks from a 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.

Given these threats and resource constraints, our approach for securing WSNs concedes that a well-equipped intruder can compromise individual sensor nodes, but that the overall design of our secure routing system should tolerate these intrusions such that the network as a whole remains functioning. We assume that the base station has considerably more resources to defend itself against attacks, and therefore concentrate on securing the system against attacks on the weakest links, namely the resource-poor sensor nodes. We have designed and implemented an INtrusion-tolerant routing protocol for wireless Sensor Networks (INSENS) [14] that has the property that a single compromised node can only disrupt a localized portion of the network, and cannot bring down the entire sensor network.

The INSENS secure routing system adheres to the following design principles. First, to prevent DOS-style flooding attacks, individual nodes are not allowed to broadcast to the entire network. Only the base station shown in Figure 1 is allowed to broadcast. The base station acts as a gateway to the wired world, e.g. a satellite uplink connecting to terrestrial networks. The base station is loosely

authenticated via a one-way sequence number, so that individual nodes cannot arbitrarily spoof the base station and thereby flood the network. Sensor nodes are restricted to only unicasting a packet, and then only to the base station, thereby preventing DOS/DDOS broadcast attacks. Peer-to-peer sensor communication is not directly supported, though tunneling through the base station permits indirect sensor-to-sensor communication. 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 knowledge of the topology that is correct, though it may only represent a partial picture due to malicious packet dropping. Third, to address resource constraints, 1) symmetric key cryptography is chosen for confidentiality and authentication between the base station and each resource-constrained sensor node, since it is considerably less compute-intensive than public key cryptography, and 2) the resource-rich 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, as shown in Figure 1. The goal is to have disjoint paths so that even if an intruder takes down a single node or path, secondary paths will exist to forward the packet to the correct destination.

In the remainder of the paper, we provide an overview of the INSENS system in Section 2, present simulation results in Section 3, an implementation of INSENS in Section 4, and address related work in Section 5.

## 2 Protocol Description

In this section, we provide a brief overview of INSENS. For a more detailed description, see [14]. INSENS is comprised of a route discovery phase and a data forwarding phase. The route discovery phase ascertains the topology of the sensor network and builds appropriate forwarding tables at various nodes. Route discovery is subdivided into 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, each sensor node send its neighborhood topology information back to the base station using a *feedback message*. In the third round, the base station authenticates the neighborhood information, constructs a topological picture of the network, computes the forwarding tables for each sensor node, and sends the tables to the respective nodes using a *routing update message*. The data forwarding phase enables forwarding of data from each sensor node to the base station, and vice versa. A symmetric communication channel is assumed, i.e. if node  $a$  can hear a message from node  $b$ , then  $a$  can send a message to  $b$ .

Each node has a shared symmetric key with base station. Every node also possesses a globally known one-way function  $F$  and initial sequence number  $K_0$ .  $F$  and  $K_0$  are used together to loosely authenticate messages from the base station, as explained next. All three pieces of information, namely  $F$ ,  $K_0$ , and the shared symmetric key, are distributed in advance, i.e. they are preprogrammed

into each sensor node before deployment. We envision that military applications will for example permit secret keys to be preprogrammed into sensor nodes before deployment.

## 2.1 Route Discovery: Route Request

The base station initiates the first round whenever it needs to construct the forwarding tables of all sensor nodes. The base station broadcasts a *request message* that is received by all of its neighbors. 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 a duplicate request message, the identity of the sender is added to its neighbor set, but the request is not rebroadcast.

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, the base station uses a one-way cryptographic hash function  $F$  to generate a sequence of numbers  $K_0, K_1, \dots, K_n$ , such that  $K_i = F(K_{i+1})$ , where  $0 \leq i < n$ . Initially, every node knows  $F$  and  $K_0$ . In the first route discovery phase, the base station includes  $K_1$  in the request message that it broadcasts. In general, the base station uses  $K_i$  in the  $i^{\text{th}}$  route discovery phase. Each node can verify that the sequence number did indeed originate from the base station by computing  $K_i = F(K_{i+1})$ . An attacker who compromised a sensor node would be unable to guess the next one-way sequence number given the most recent sequence number, i.e. given  $F$ ,  $K_0$ , and the most recent sequence  $K_i$ , the attacker cannot invert  $F$  to generate the next sequence number  $K_{i+1}$ . As a result, a compromised node cannot spoof the base station by generating new sequence numbers. However, a compromised node could repeat the current sequence number in a request message to its downstream nodes, who would then believe that the compromised node is the base station. The damage in this case is localized to the compromised node, which was our design objective. The rest of the network will receive the authentic base station's route request first, and will therefore ignore the compromised node's route request. Our usage of one-way functions leverages the approach taken by the  $\mu$ TESLA protocol [10], but differs in the sense that the numbers in the one-way chain are sequence numbers rather than symmetric keys.

The second mechanism that we use is a keyed MAC algorithm. Each sensor node is configured with a separate secret key that is shared only with the base station. When a node  $x$  receives a request message for the first time, it appends its identity to the path list, and then generates a MAC of the complete new

path with its key. This MAC is also appended to the request message, before the modified request message is forwarded downstream. This MAC will eventually be used by the base station to verify the integrity of the path contained in the packet. Also, when a node is compromised, only one secret key is revealed, so an attacker cannot compromise the entire network.

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.

## 2.2 Route Discovery: 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  $x$  has forwarded its request message in round one, it waits for a certain timeout interval before generating a feedback message. During this time interval, it listens to the local broadcasts from neighboring nodes forwarding the same request message, and stores the neighbor's identity and the neighbor's MAC embedded within the request message. After the timeout, the sensor node will send its list of neighbors (upstream, peer, and downstream) back to the base station, where each neighbor is identified by the neighbor's identity and the neighbor's MAC. The sensor node applies its keyed MAC to the topology data, i.e. the list of neighbors, to further protect the integrity of the feedback message. The messages that reach the base station are guaranteed after verification to be correct and secure from tampering.

Routing of the *feedback message* from a node  $x$  to the base station follows the reverse path taken by the request message that initiated the feedback response. To ensure that malicious nodes do not generate false paths while forwarding a feedback message, a node places its parent identification information along with its parent's MAC, that it received in the first request message. Each node will choose one legitimate upstream parent, forming a parental chain of nodes back to the base station. A compromised node will at most be able to flood each of its parents' chains back to the base stations, but no other nodes. This localizes the effect of an attack. To further restrict attacks, rate control is applied at each node; regardless of the incoming traffic rate, the outgoing traffic rate of each node is restricted to some maximum rate, thereby preventing flooding. Also, each node encrypts appropriate information in the feedback message it sends to provide confidentiality against eavesdropping by a malicious node.

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 message* or by modifying the neighborhood information of nodes, which can be detected at the base station. These 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 affect only a limited number of upstream nodes.

### 2.3 Route Discovery: Computing and Propagating Multipath Routing Tables

After sending its 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*. Each node returns an authenticated list of its neighboring nodes. As a result, the base station is able to verify the neighbor information and detect tampering with feedback messages. The base station constructs a topology of the network from these authenticated feedback messages, though this picture of the network may be incomplete due to dropped feedback messages. From this connectivity information, the base station computes the forwarding tables of each node in the network.

INSENS incorporates redundancy in routing by building multiple redundant paths to bypass intruders while routing messages, as shown in Figure 1. 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. 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 will receive at least one copy of the message that has not been tampered with.

While INSENS is largely agnostic to the particular criteria for choosing multiple paths, we chose the following multipath heuristic in order to proceed with our implementation of INSENS. For a sensor node  $A$ , the first path from  $A$  to the base station is chosen using Dijkstra's shortest path algorithm. To determine the second path, three sets of nodes,  $S_1$ ,  $S_2$ , and  $S_3$  are first constructed.  $S_1$  is the set of nodes belonging to the first path,  $S_2$  is the set of nodes belonging to  $S_1$  and any neighbor nodes of the nodes in  $S_1$ , and  $S_3$  is the set of nodes belonging to  $S_2$  and any neighbor nodes of the nodes in  $S_2$ . All three sets exclude  $A$  or the base station. The second path is then computed as follows:

1. Remove all nodes in  $S_3$  from the network, and find the shortest path from  $A$  to the base station. If such a path is found, terminate the computation. The path found is the second path.
2. Otherwise, remove all nodes in  $S_2$  from the original network. Find the shortest path from  $A$  to the base station. If such a path is found, terminate the computation. The path found is the second path.
3. Remove all nodes in  $S_1$  from the original network. Find the shortest path from  $A$  to the base station. If such a path is found, it is the second path. Otherwise, there is no second path from  $A$  to the base station.

Notice that 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. Finding a better algorithm to compute multiple paths in INSENS is part of our future work.

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 a 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 [10] can be used to preserve the authentication, integrity, and confidentiality of the forwarding tables.

## 2.4 Data Forwarding

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:  $\langle \textit{destination}, \textit{source}, \textit{immediate sender} \rangle$ . *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  $D$ :  $S \rightarrow a \rightarrow b \rightarrow c \rightarrow D$ , the forwarding table of node  $a$  will contain an entry  $\langle D, S, S \rangle$ , forwarding table of  $b$  will contain an entry  $\langle D, S, a \rangle$ , and the forwarding table of  $c$  will contain an entry  $\langle D, S, b \rangle$ . 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  $\langle \textit{destination}, \textit{source}, \textit{immediate sender} \rangle$  in its forwarding table. If it finds a match, it forwards (broadcasts) the data packet.

## 3 Simulation

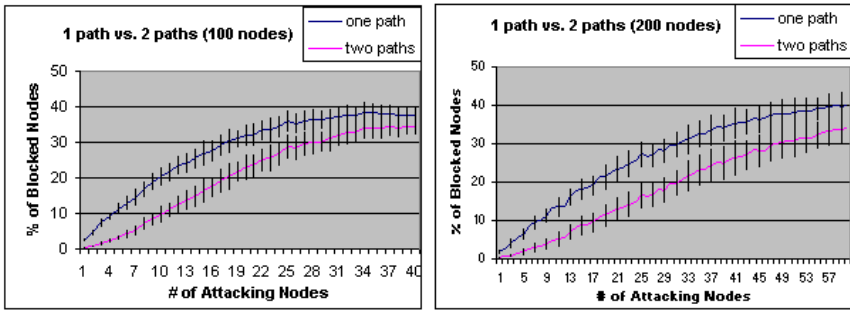
We have simulated INSENS on nsclick [16], a network simulation tool that combines the ns-2 network simulator with the Click Modular Router. We implemented our own Click element to simulate the behavior of INSENS on sensor nodes and the base station. Ns-2 was used to simulate the wireless network environment, including the MAC (Medium Access Control) protocol and the lower layers of the wireless network, as well as the geographic distribution of nodes.

### 3.1 Malicious Attack during Data Forwarding

INSENS builds two 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 2 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  $1500 \times 1500m^2$  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.



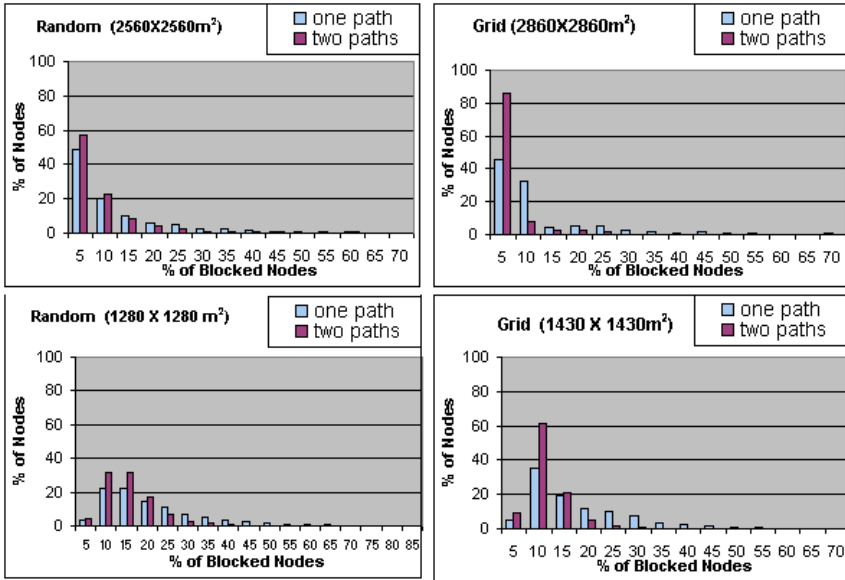
**Fig. 2.** Multi-node attack on a sensor network that has secure single path and multipath routing. Left graph shows 100 nodes, and right graph shows 200 nodes. X axis: #of attacking nodes. Y axis: #of blocked nodes unable to send packets.

### 3.2 DOS Attacks

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 3 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 multi-path 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 for each random topology 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 grid is a  $14 \times 14$  square.



**Fig. 3.** Histograms of simulated DOS attacks for sparse and dense random and grid topologies.

Figure 3 reveals the performance of INSENS against a single node launching a DOS attack. For either uniform grids or random positioning, we first generate a given topology of scattered nodes. For this topology, we let each node at a time become a DOS intruder and measure the number of blocked nodes downstream affected by the DOS intruder. This generates a histogram per topology. The x-axis records the percentage of nodes that may be blocked by a single-node DOS attack, and the y-axis records the percentage of such nodes in the topology who, if they turned malicious, would have the power to block the number of nodes listed in the x-axis. For clarity, we have grouped the x-axis into bins of 0-5 %, 6-10 %, etc. For both random and grid topologies, we generate 50 such topologies and plot the averaged histogram shown above.

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 multi-path algorithm provides better protection against DOS attacks than the single path approach. The multi-path approach performs far better for the grid topology, because the grid nearly always offers a valid redundant second path. The best performance of the multi-path 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 multi-path approach is obtained for sparse random topologies (upper left graph), in which nodes have few neighbors and few alternate paths (usually only one path) to the base station. In this case, the multi-path approach performs only slightly better than single path routing.

As the network becomes denser, moving from the top row of graphs to the bottom row in Figure 3, 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.

While the figures measure the average response of INSENS, an attacker would benefit by exploiting the topology's structure and identifying the weakest nodes that would partition the graph. Such a partitioning attack would be largely ineffective in grids and/or dense topologies, because such topologies do not easily partition because of alternate paths. Partitioning is a more effective attack for topologies that are both random and sparse. We have not specifically measured INSENS's performance against such a partitioning attack.

## 4 Implementation

In our implementation, we use UC Berkeley MICA sensor motes [13] as the sensor nodes. The program runs on Atmel Atmega128 microcontroller. The motes support a 4MHZ processor with 128K Bytes code memory and 4K Bytes internal data memory, and an RFM Monolithics TR 1000 radio at 19.2Kbps. INSENS is running on TinyOS 1.0, which is a small, open source, event-driven, energy efficient operating system developed for sensor networks at UC Berkeley.

### 4.1 Cryptographic Algorithm

To implement INSENS on motes, we need to choose a secure, efficient cryptographic algorithm that can operate correctly, given the resource constraints of motes. To save memory, we should reuse a single cryptographic algorithm for data encryption, MAC generation, and one-way sequence number, as long as their implementations are secure. We chose RC4, RC5, and Rijndael (AES) as candidates. RC5's implementation varies according to the number of rounds. More rounds result in higher security, but require more resources. We implemented RC5 with 5 rounds and 12 rounds. The output of 5 rounds is statistically no different from a random number, and 12 rounds is recommended by

Rivest [17]. We tested the performance of a stream cipher RC4 to compare its performance with block cipher algorithms. RC4 is a very fast stream cipher, but has some weaknesses when used in wireless networks. [18] We also implemented Rijndael on the motes and compared its performance with RC5. We used a standard version of Rijndael [19]. It uses about 1KB memory. There is a fast version that uses about 4KB lookup tables, but that exceeds the memory capabilities of the mote.

To measure performance, we implemented RC4, RC5, and Rijndael on motes to encrypt  $200 \times 128$  bits of data with CBC mode. To measure the speed of these algorithms on motes, we let the base station send a “*begin*” signal to a mote. On receiving this signal, a mote begins its computation, and after completing the computation, it sends back the result to the base station. The base station records the time interval between when it sent the signal and when it got the data back, verifies the result, subtracts the round-trip time (which is measured in the same way without the mote doing any encryption), and gets the computing time. For each algorithm, we tested it for 20 times. Table 1 shows the calculated average time for computing 128 bits of data for each algorithm.

From Table 1 we see that: 1) RC5 is a good candidate for motes. It uses less memory (both in code size and data size), and it is very efficient. 2) Compared with RC5, Rijndael is very slow. Based on our result, to encrypt a 30Bytes packet, it would spend about 0.2 seconds. However, we believe that in the near future, as sensor nodes become faster and acquire more memory, Rijndael will become a good candidate for cryptographic algorithm on sensor networks. In our implementation, we used RC5 with 5 rounds. We think it is good enough for sensor networks. We can also use RC5 with 12 rounds.

**Table 1.** Cryptographic Algorithm Overhead

|                    | RC4   | RC5      |           | AES     |
|--------------------|-------|----------|-----------|---------|
|                    |       | 5 Rounds | 12 Rounds |         |
| Speed (128bits/ms) | 1.299 | 5.471    | 12.475    | 102.483 |
| Data Size (B)      | 258   | 68       | 124       | 1165    |
| Code Size (B)      | 580   | 1436     | 1436      | 9492    |

We have also implemented RSA public key cryptography on the mote platform and report the following preliminary results. We decrypted 64 bytes of data on the mote with a 1024-bit RSA public key. We found that the measured delay for decryption was approximately 15 seconds. This suggests that public key cryptography could be used in a limited way, e.g. for symmetric key exchange, for certain sensor networks. We also attempted to implement encryption with an RSA private key on the mote, but found that the encryption code died during execution. We hypothesize that encryption exceeded the mote’s memory capacity, since RSA encryption consumes more memory than decryption, though more tests are needed to confirm this hypothesis.

**Message Authentication Code generation.** MAC plays a critical role in INSENS. It is used to authenticate each node, its path, and its neighbor information. We use standard CBC mode to generate MAC with block cipher RC5 [20].

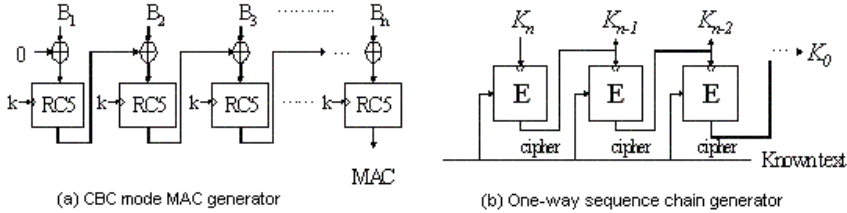


Fig. 4. CBC-based MAC generation

**One-way sequence number generation.** The one-way sequence number is used to loosely authenticate the base station. To generate the one-way sequence number, we need a secure one-way function. Our approach is based on the following criteria: By knowing a plaintext and the corresponding ciphertext computed using a block cipher algorithm, such as RC5, we cannot know the key that was used to generate the ciphertext. Our one-way sequence number generator is shown in Figure 4(b). The base station chooses a random key  $K_n$  and uses it to encrypt a well-known plaintext and gets a cipher. This cipher is  $K_{n-1}$  and the base station uses it as a key to encrypt the same known plaintext. This process continues until we get  $K_0$ .

## 4.2 Implementation Issues

**Base Station and Node.** We implemented base station in Java. The base station gets information from the mote on the programming board and processes the information, and sends routing tables back to each mote. In our implementation, we used the same strategy described in [14] to find two paths for each node. But we choose BFS (Breadth First Search) algorithm instead of Dijkstra because we assume the cost of each link is same. We implemented INSENS on TinyOS 1.0 with NesC. All of our computing intensive functions are written as tasks, to prevent them from blocking packets or timer interrupts.

**Feedback Message Segmentation.** On the current TinyOS, the default packet size is 30 bytes, though this can be modified. However the feedback message of INSENS can be far longer, because it contains an authenticated list of neighbors. In our implementation, we segment one feedback message into multiples of 30 byte feedback packets. We add two constraints for feedback packet

segmentation to make it work with INSENS and prevent possible attacks. 1) Every segment packet has a sequence number. Any node must forward lower sequence packet before forwarding a higher sequence packet. When a node gets a higher sequence packet while it hasn't got a lower sequence packet, it must drop that packet. 2) The whole path information must be put in the first packet. Upstream nodes need it to forward packets. That limits the longest path at 9. This is suitable for a moderately sized network. Because every feedback message contains a MAC number, which is generated by CBC mode, the malicious node cannot change the sequence of segment packet, or replace a segment packet. The base station can verify the integrity of feedback message sequence packets with the MAC.

**Packet Loss.** During our experiments, we found that there were many packet losses. The reasons for this may be: 1) The MAC (media access control) layer of TinyOS cannot deal with loss of packets, and INSENS needs to send lots of packets. 2) The packet sending/receiving components of TinyOS cannot receive packets in time.

We employed the following methods to alleviate packet loss. First, random delay is introduced in each mote before forwarding to reduce collisions. Second, when a mote gets a packet, it copies the packet to its frame variable immediately. With these mechanisms, the packet loss was significantly reduced. We note improved MAC protocols [15] could be adopted in the future.

### 4.3 Performance Evaluation

We have implemented INSENS on motes to build 3-node, 6-node and 10-node networks. Figure 5(a) shows the network topology setup by INSENS for a network of 6 nodes. Every node has its own routing table to route packets. We see that the node 5 has two paths to base station, the first goes through node 6, the second traverses nodes 4 and 1. Because of packet losses, the base station cannot obtain complete network topology information, yet it can still build part of the network based on the request and feed-back messages that do arrive. This is an important feature of INSENS. We measured the memory usage of INSENS and total time to setup the whole network with INSENS, to assess the practicality of INSENS.

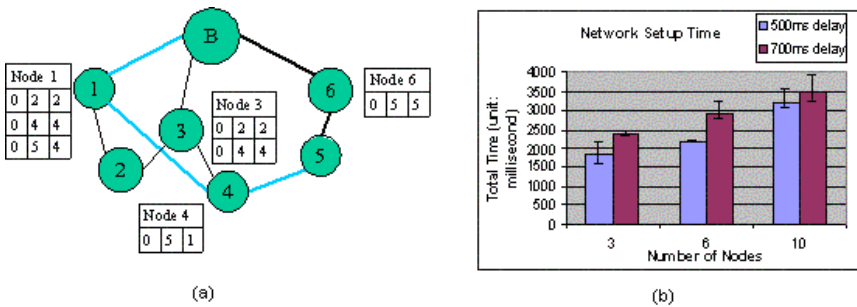
**Memory Usage of INSENS on Motes.** Table 2 shows the memory usage of INSENS. "*Feedback*" is for saving the whole feedback message before segmenting it. "*Packet*" is for saving the incoming packets. In our implementation, we didn't focus on saving memory space, but the result shows that the memory requirements of INSENS can be easily satisfied by the constraints of current mote-based sensor networks. Additional memory savings could be achieved. For example, with a good packet processing mechanism, we don't need "*packet*" space, and with a better packet segmentation implementation, we don't need "*feedback*".

**Table 2.** Memory Consumption of INSENS (Unit:byte)

| code  | total data | Crypto | neighbor info | msg & MAC | feedback | packet | OS and others |
|-------|------------|--------|---------------|-----------|----------|--------|---------------|
| 19000 | 1200       | 68     | 105           | 105       | 200      | 360    | 360           |

**Network Setup Time.** In our implementation, the base station broadcasts a request message, receives all feedback messages, and computes the routing tables. It sends each node’s routing table, and waits for a “*routing table received*” message from every node. We measure the time interval between the time the base station broadcasts its request message and the time it gets all “*routing table received*” messages. We set the network as a dense network, so every node has several neighbors. As the number of nodes increased, we experienced more packet losses. But because of the redundancy in neighbor information, the base station was usually able to setup the network based on the limited number of feedback messages that did arrive.

There are several factors affecting the setup time: 1) execution time of cryptographic algorithm, 2) execution time of packet processing, such as sending, receiving, copying, and routing, and 3) waiting time in INSENS, that includes random delay, feedback message waiting time, and the base station waiting time. The base station waits at most 500 ms after receiving a feedback packet. This wait time is reset with each new feedback message. Eventually, no more feedback messages will arrive and the base station will timeout and move on to computing the routing tables. Each sensor node also waits at most 500 ms for neighbor information to be collected. We also tested 700 ms timeouts for the sensor nodes only (not base station). The base station unicast a custom routing table to each mote, and waits 100 ms between sending each routing table. We found that the total network setup time is dominated by the waiting time of the sensor nodes. In comparison, the computation time of RC5-based cryptographic algorithms is relatively short. Figure 5(b) shows our aggregate test results.



**Fig. 5.** (a) Routing tables built by INSENS (b) Network setup time

## 5 Related Work

Sensor network security is a critical issue in sensor network research [4]. Ganesan et al propose a redundant “multipath” routing approach for a sensor network [5] in order to provide fault tolerance and reliable data dissemination. INSENS is largely agnostic to the particular multipath approach employed.

In the field of ad hoc wireless networking, previous work on secure routing employs public key cryptography to perform authentication [6] [4] [7] [8] [9]. Unfortunately, resource constraints in sensor network limit the applicability of these current public/asymmetric key standards.

SPINS [10] addresses secure communication in resource-constrained sensor networks, introducing two low-level secure building blocks, SNEP and  $\mu$ TESLA. Our work uses ideas from SNEP and  $\mu$ TESLA to build INSENS. Like  $\mu$ TESLA, we employ one-way functions, but differ in the sense that the numbers in the one-way chain are sequence numbers rather than symmetric keys. In addition, we are not constrained by time synchronization or a delayed release schedule.

SEADS [11] and Ariadne [12] use symmetric cryptography, a one-way hash function, TESLA, and MACs to build secure wireless network routing. INSENS differs in that it focuses on an asymmetric or hierarchical architecture with a base station and sensors, rather than on peer-to-peer routing.

Staddon et al [21] proposes an efficient algorithm to trace failed nodes in sensor network. Their work also puts intensive computing on the base station, and employs route discovery in a manner similar to our first two rounds. The paper does not address the issue of compromised nodes.

## 6 Conclusions

In this paper, we have provided an experimental evaluation of INSENS, which is an intrusion-tolerant routing protocol for wireless sensor networks. The resilience of INSENS’s multipath performance against various forms of communication-based attacks by intruders is evaluated in simulation. The paper describes practical experiences with implementations of RC5 and AES encryption standards on motes, an RC5-based scheme to generate message authentication codes (MACs), and an RC5-based generation of one-way sequence numbers.

## References

1. Wood, A., Stankovic, J.: Denial of Service in Sensor Networks, *IEEE Computer*, Oct 2002, pp. 54–62.
2. Slijepcevic, S., Potkonjak, M., Tsiatsis, V., Zimbeck, S., Srivastava, M.: On Communication Security in Wireless Ad -Hoc Sensor Networks, *Eleventh IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE’02)*, pp. 139–144.
3. Karlof, C., Wagner, D.: Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures, *First IEEE International Workshop on Sensor Network Protocols and Applications*, May 2003.

4. NAI Lab: [http://www.nai.com/nai.labs/asp\\_set/crypto/crypt\\_senseit.asp](http://www.nai.com/nai.labs/asp_set/crypto/crypt_senseit.asp).
5. Ganesan, D., Govindan, R., Shenker, S., Estrin, D.: Highly Resilient, Energy Efficient Multipath Routing in Wireless Sensor Networks. *Mobile Computing and Communication Review (MC2R)* Vol 1., No.2. 2002.
6. Kong, J.J., Zerfos, P., Luo, H., Lu, S., Zhang, L.X.: Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks. *International Conference on Network Protocols (ICNP 2001)*.
7. Papadimitratos, P., Haas, Z.: Secure Routing for Mobile Ad hoc Networks. *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*.
8. Zhou, L., Haas, Z.: Securing Ad Hoc Networks. *IEEE Network Magazine*, vol. 13, no.6, November/December 1999.
9. Zhang, K.: Efficient protocols for signing routing messages. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS '98)*, San Diego, California, March 1998.
10. Perrig, A., Szewczyk, R., Wen, V., Culler, D., Tygar, J.D.: SPINS: Security Protocols for Sensor Networks. *Proceedings of Seventh Annual International Conference on Mobile Computing and Networks MOBICOM 2001*, July 2001.
11. Hu, Y., Johnson, D., Perrig, A.: SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. *Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '02)*.
12. Hu, Y., Perrig, A., Johnson, D.: 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)*.
13. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System architecture directions for network sensors. *ASPLOS 2000*
14. Deng, J., Han, R., Mishra, S.: INSENS: Intrusion-tolerant routing in wireless Sensor Networks. *Technical Report CU-CS-939-02*, Department of Computer Science, University of Colorado, November 2002.
15. Ye, W., Heidemann, J., Estrin, D.: An Energy-Efficient MAC Protocol for Wireless Sensor Networks. *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, New York, NY, USA.
16. Neufeld, M., Jain, A., Grunwald, D.: Nsclck: Bridging Network Simulation and Deployment. *MSWiM'02*, September 28, 2002. Atlanta, Georgia, USA.
17. Schneier, B.: *Applied Cryptography*, second edition. John Wiley & Sons, Inc. 1996
18. Borisov, N., Goldberg, I., Wagner, D.: Intercepting mobile communications: The insecurity of 802.11. In *Proceedings of MOBICOM 2001*.
19. Daemen, J., Rijmen, V.: *The Design of Rijndael*. Springer, (2001) 221–227
20. Menezes, A., etc: *Handbook of Applied Cryptography*. CRC Press, (1996) 353–354.
21. Staddon, J., Balfanz, D., Durfee, G.: Efficient Tracing of Failed Nodes in Sensor Networks. *First Workshop on Sensor Networks and Applications, WSNA'02*, Atlanta, Georgia, USA.