Intrusion Tolerance and Anti-Traffic Analysis Strategies For Wireless Sensor Networks

Jing Deng Richard Han Shivakant Mishra
Computer Science Department
University of Colorado at Boulder
Boulder, Colorado, USA
{jing,rhan,mishras}@cs.colorado.edu

Abstract

Wireless sensor networks face acute security concerns in applications such as battlefield monitoring. A central point of failure in a sensor network is the base station, which acts as a collection point of sensor data. In this paper, we investigate two attacks that can lead to isolation or failure of the base station. In one set of attacks, the base station is isolated by blocking communication between sensor nodes and the base station, e.g. by DOS attacks. In the second attack, the location of the base station is deduced by analyzing data traffic towards the base station, which can lead to jamming and/or discovery and destruction of the base station. To defend against these attacks, two secure strategies are proposed. First, secure multi-path routing to multiple destination base stations is designed to provide intrusion tolerance against isolation of a base station. Second, anti-traffic analysis strategies are proposed to help disguise the location of the base station from eavesdroppers. A performance evaluation is provided for a simulated sensor network, as well as measurements of cryptographic overhead on real sensor nodes.

1. Introduction

Wireless sensor networks (WSNs) are rapidly growing in their importance and relevance to both the research community and the public at large. A distributed wireless sensor network is formed by a large number of tiny and inexpensive sensor nodes. These nodes are typically resource-constrained, with limited energy lifetime, low-power microsensors and actuators, slow embedded processors, limited memory, and low-bandwidth radios. For example, both sensor motes [13] and nymphs [2] contain a 4 MHZ processor, 4 KB SDRAM memory and 128 KB flash memory to run an operating system and application programs. Additional storage of 4-512 KB EEPROM is available to save sensed data. The Chipcon CC1000 radio operates at a rate of 19.2 kbps.

The sensor nodes self-organize into a multi-hop wireless network that collects and forwards sensor data to an information sink, usually a base station acting as a gateway to the wired Internet. The structure of a typical wireless sensor network is illustrated in Figure 1. In general, the computing resources of each base station are much greater than the computational abilities of the sensor nodes. The large number of sensor nodes and the small number of base stations collectively form an asymmetric and hierarchical wireless sensor network. Applications of WSNs are rapidly emerging and have become increasingly diverse, ranging from habitat monitoring [18] to indoor sensor networks with sensor-enabled user interfaces [6] to battlefield monitoring [3] and seismic monitoring of buildings.

In certain WSN applications, such as home security monitoring or military deployments, security, fault tolerance, and intrusion tolerance are especially important. Intrusion tolerance has been studied in the context of wired networks [5][20][21][23]. However, wireless sensor networks face a combination of threats that are not normally faced by wired networks. First, the broadcast nature of the wireless communication medium significantly enhances the capabilities of an adversary to eavesdrop, tamper with transmitted packets, and inject packets to initiate denial-of-service (DOS) attacks. These susceptibilities also apply to wireless LANs such as 802.11 and mobile ad hoc networks. Second, WSNs are highly resource constrained, which has strong implications on the type of cryptography that can used in sensor nodes, e.g. lightweight symmetric key cryptography such as RC5 has been shown to be effective [19], while compute-intensive public key cryptography such as RSA is infeasible at present[8]. The relatively weak defenses of sensor nodes are susceptible to external attacks by much stronger adversaries equipped with more powerful computing and communication equipment. Third and perhaps the most unique, sensor nodes are distributed in the field in-

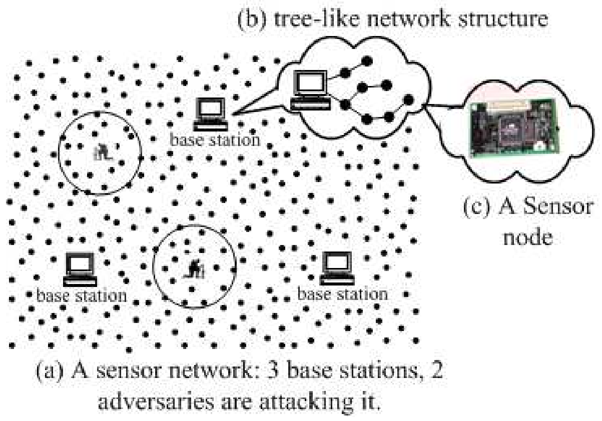

(b) tree-like network structure
base station
(c) A Sensor node
base station
base station
(a) A sensor network: 3 base stations, 2 adversaries are attacking it.

Figure 1. An example of a typical wireless sensor network.

situ and therefore lack the physical security of most other forms of wired and wireless networking. As a result, WSNs are highly susceptible to the physical compromise of one or more sensor nodes. Once compromised, the sensor node(s) can be exploited by an intruder to damage the WSN through DOS, jamming, and spoofing attacks.

This paper focuses on improving the intrusion tolerance of a WSN against attacks focused on isolating or destroying the base station. As shown in Figure 1, the base station is a central point of failure. If an adversary can successfully attack the base station, then the adversary can largely disable the WSN. The variety of attacks that can be mounted against the base station include remote DOS attacks from deep within the WSN that flood the base station with packets, remote spoofing of the base station to misdirect legitimate sensor data and thereby starve the base station, and eavesdropping to deduce the vicinity of the base station so that it can be locally jammed or even physically destroyed if its precise location can be discovered. The DOS, spoofing and jamming attacks all result in isolation of the base station. Despite the best electronic countermeasures, an adversary may get lucky and destroy a single base station.

To address these kinds of remote and local attacks upon a WSN's base station, this paper develops two security strategies. First, a key focus of this paper is on developing techniques that can limit the damage from disrupting the communication between base stations and sensor nodes. In particular, we introduce mechanisms that enable the secure set up of multiple routing paths to multiple base stations. With this scheme, even though an adversary can attack and destroy part of the sensor network, e.g. isolate a minority of base stations, the rest of the network can survive and continue to report data. A second key focus of this paper is on introducing novel techniques that protect the location and identity of the base station from being easily discovered. For example, if an attacker is able to snoop on packet traffic, and knows that all sensor packets are routed towards the base station, the attacker could follow packets, gradually trace the route back to the base station and thereby discover the vicinity of the base station for local attacks. This paper proposes a variety of techniques to combat such traffic analysis attacks, and thereby improve the intrusion-tolerance properties of WSNs. The net effect of both secure multipath routing setup and anti-traffic analysis is to improve the intrusion tolerance of WSNs to base station-focused attacks.

2. Network Framework and Threat Model

We build our security schemes based on the common sensor network structure described in TinyOS [13] and TinyDB [17]. We assume that the sensor nodes are organized in a tree-like network routing structure around each base station, as shown in figure 1 (b). Each base station is the root node of a tree. Every sensor node is a node in some of the trees. Each node has a number of child nodes that are its downstream nodes, and a parent node that is its upstream node. Every sensor node processes the sensed data from all of its child nodes and itself, and sends the result to its parent node. Each node has its activity range v: if the distance between two sensor nodes is no more than v, the pair of nodes can send and receive data to and from each other.

For the capabilities of an adversary, we assume that:

- An adversary can capture sensor nodes and is capable of compromising a sensor node to obtain all of its information, e.g. symmetric keys. In addition, an adversary can reprogram a sensor node to convert it into a malicious node. But we assume that the adversary needs some time to compromise a node.
- An adversary has a jamming range d, $d \geq v$. Within d, an adversary can generate radio signals to interfere with signals generated by sensor nodes or base stations.
- An adversary can receive any data from any sensor node or base station, if the distance is less than v. We assume that an adversary's packet acceptance range is still v. Although it is easy to send a stronger data signal to a larger range than a normal sensor node's range, it is difficult to receive data from a sensor node that is further than v, since it needs needs very sensitive, and expensive equipments.
- An adversary can physically move from place to place.
- However, an adversary doesn't have global information about the whole network, and cannot jam the entire network. Suppose the whole sensor network has range D. Then we assume $d \ll D$.

Our assumption is that it is very difficult for an adversary to obtain sufficient global information to destroy the entire sensor network. Instead, the adversary is assumed to have limited local knowledge of the sensor network. Moreover, we assume that a resource-rich base station has sufficient functionality to protect itself from tampering and is resistant to observations via camouflage. As a result, an adversary's only threat options to the base stations are to jam the communication medium, destroy the base station, spoof the base station, or flood the base station.

3. Redundant Paths Setup

To provide intrusion tolerance and fault tolerance, we introduce redundancy in the form of multiple base stations. Since an adversary can obstruct delivery of sensor data that is routed over only one path to one of the base stations, we introduce multi-path routing redundancy to improve intrusion tolerance of WSNs. Ensuring that each path is routed towards a different base station can further improve the effectiveness of this approach. Our contribution in the following is to describe how a multi-path multi-base station routing scheme can be constructed in a WSN while still limiting the ability of an adversary to spoof a base station and/or launch DOS attacks against a base station.

3.1. Securing Multi-Path Multi-Base Station Routing

The simplest way to set up multiple paths for each sensor node to multiple base stations is to use a flooding message: each base station broadcasts a unique request message (called the REQ message). When a sensor node first receives the REQ message from a base station, it records the sender of the packet as its parent node for that base station, and re-broadcasts the REQ message to its neighbor and child nodes. This sensor node then ignores all copies of the same REQ message that it received later. In this manner, the REQ message generated from each base station floods the entire network, while every node forwards that message just once, and the path between a child node and its parent node forms a tree rooted at that base station. If there are multiple base stations flooding their own REQ messages, every sensor node will have one path for each base station.

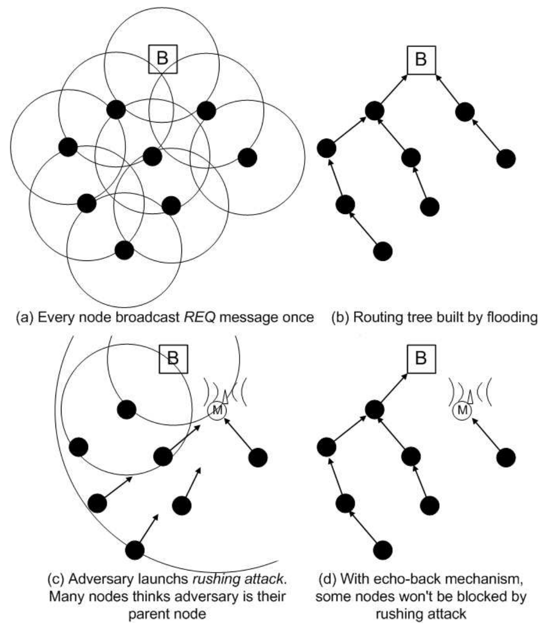
However, this simple scheme cannot prevent a malicious compromised node from spoofing a base station by sending forged REQ messages. Every node will think the forged message is generated by this base station, and will forward the forged REQ message. That message will flood the whole network, and can be repeatedly sent in a form of DOS attack. In addition, all sensor nodes will build a routing tree rooted at the malicious node. To defend against such an attack, we adapt a scheme proposed in [8] of using a one-way hash chain to lossely authenticate REQ messages. Here we briefly describe this solution.

A one way hash chain is generated by a one-way function F. F has the property that if we know x, it is easy to compute $y = F(x)$. But if we only know y, it is computationally infeasible to obtain $x = F^{-1}(y)$. A one-way hash chain is a sequence of numbers, $K_n, K_{n-1}, \ldots, K_0$, such that $K_{i-1} = F(K_i)$, where $0 < i \leq n$. Each base station α randomly selects a seed K_n^α and computes a one-way hash chain $H^\alpha =< K_n^\alpha, K_{n-1}^\alpha, \ldots K_0^\alpha >$ with function F. Each sensor node is pre-configured with the initial number K_0^α. When a base station α sends its first REQ message, that message contains a one-way hash chain number K_1^α. When a sensor node receives this message, it verifies the one-way hash chain number in the REQ message by checking if $K_0^\alpha = F(K_1^\alpha)$. If such a match is found, the sensor node assumes that the message has been generated from base station α. The node then caches the one-way hash chain number it just received, and process the message; otherwise the message is dropped. When the base station α sends ith REQ message, it attaches K_i^α. When a node gets this message, it will use its cached one-way hash chain number to verify the message by applying the function F a finite number of times until the cached one-way hash chain is encountered. The advantage of such a scheme is that even if an adversary compromises a sensor node and obtains F and the seed, it cannot generate future numbers in the one-way hash chain. In this way, the ability of an arbitrary compromised sensor node to spoof a base station by generating false REQ messages is severely limited.

However, there remains a security problem for this one-way hash chain approach. An adversary can launch a rushing attack [15] [14] to "capture" a large number of downstream nodes. In such a rushing attack, when the adversary receives a REQ message, it immediately rebroadcasts the message with much higher transmission power. The nodes "captured" within the adversary's jamming range d will be misled into thinking that the adversary is their parent node. In addition, the capture effect is magnified since the REQ message sent by the adversary reaches downstream attacked nodes earlier than normal REQ message propagation, so that the attacked nodes will further capture more of their downstream nodes. All such captured downstream nodes will fail to connect to the correct base station.

3.2. Echo-back Scheme to Identify Neighbor Nodes

3.2.1. Echo-back Process to Verify Neighbor Nodes To address the rushing attack problem, we propose the following echo-back scheme. An adversary is able to launch a rushing attack when a sensor node fails to check whether



B
(a) Every node broadcast REQ message once
(b) Routing tree built by flooding
M
(c) Adversary launchs rushing attack. Many nodes thinks adversary is their parent node
(d) With echo-back mechanism, some nodes won't be blocked by rushing attack

Figure 2. REQ message flooding, rushing attack and echo-back countermeasure.

a sender with an expanded transmission range can reciprocally receive data. We observe that if a sensor node can detect that it cannot reach the transmitter, then that node can identify and block a rushing attack. The sensor node's activity range v is smaller than the jamming range d of the adversary. We assume that the adversary can only hear data within range v, because the data sent by a sensor node is too weak to be detected beyond range v. If each sensor node constructs a set of reachable neighbor nodes, and is only willing to receive REQ messages from this set of neighbor nodes, then spoofed REQ messages from an adversary transmitting at maximum power will be ignored. Thus, the damage from a rushing attack can be restricted within a small range v.

To identify neighbor nodes, we introduce a simple echo-back approach. In its most basic form, which we shall enhance, when a sensor node $S1$ receives a broadcast REQ message from another node $S2$, it sends an *echo* message to that node $S2$ and waits for the replies to that message. Until it receives a $feedback$ message from $S2$, the earlier broadcast message is not processed by $S1$. If a node receives the $feedback$ message from a neighbor node, it will record that node as its *verified neighbor*. To reduce delay in broadcasting, sensor nodes can run the echo-back procedure with its neighbor nodes before base stations flood their REQ messages. Thus, when a node receives a REQ message, it can immediately check if the message sender is its neighbor node. Figure 2 shows the REQ flooding scheme, the rushing attack, and the echo-back defense.

The rushing attack is not completely precluded with the echo-back defense. Multiple adversaries can cooperatively form a relay path that is shorter than the normal REQ propagation path. However, such a cooperative attack is more difficult to mount than the rushing attack addressed here.

3.2.2. Cluster Key Set Up It is useful to encrypt each REQ message at each forwarding hop, instead of sending a plaintext broadcast message. If the adversary doesn't know the key to decrypt a REQ packet, then it cannot launch a rushing attack.

To encrypt the REQ message, first each pair of verified neighbor nodes sets up a pair-wise key. The key set up is combined with the echo-back scheme. Consider first a simple pair-wise key set up in which we assume that all nodes in the network share a global key. The following process shows how to run echo-back and set up pair-wise keys between neighbor nodes.

First, every node a locally broadcasts an $echo$ message to its neighbor nodes with format:

$$echo : E_{global\_key}(ID_a||nonce)$$

Where ID is the ID of sensor node a, $nonce$ is a random number.

If node b receives this message, it generates a random number $K_{b,a}$ as the pair-wise key between a and b, and unicasts back the message with format

$$back : E_{global\_key}(ID_b||nonce+1||K_{b,a})$$

When node a receives this message, it records node b as its verified neighbor, and it compares its ID number with b's ID number. If $ID_a < ID_b$, node a and b use the random number $(K_{a,b})$ generated by a as their pair-wise key. Otherwise, if $ID_a > ID_b$, then they use the random number $(K_{b,a})$ generated by b as their pair-wise key.

The global key is only used to encrypt the pair-wise key during data transmission. If an adversary obtains the global key after a node has received its pair-wise key, then the adversary cannot know the pair-wise key. If an adversary obtains the global key before the echo-back process finishes, he can obtain the pair-wise keys within his range, but is unlikely to obtain the pair-wise keys outside of his range, because those nodes would have finished their echo-back scheme.

Recently, several random key pre-distribution schemes have been proposed to set up pair-wise keys between neighbor nodes in sensor network [10][7][9][16]. These schemes provide stronger security protection than the global key approach. We can use any of these schemes to set up pair-wise keys and verify neighborhood relationships.

After a node s has set up pair-wise keys with all of its neighbors, we propose that it sets up a single cluster key for

```
REQ_process(Packet $p$) {
  $src\_id \leftarrow p.ID_s$
  if ($src\_id \in neighbors\_set$) {
    $content \leftarrow D_{K_{src\_id}}(p.content)$
    // $p.content == E_{KC_s}(OHC||ID_B)$
    $bs\_id \leftarrow content.ID_B$
    $tmp\_ohc \leftarrow content.ohc$
    for ($i \leftarrow 0$;$i < threshold$;$i++$) {
      if ($ohc[bs\_id] == F(tmp\_ohc)$) {
        $ohc[bs\_id] \leftarrow content.ohc$
        $p.content \leftarrow E_{K_{my\_id}}(content)$
        $p.ID_s \leftarrow my\_id$
        send $p$
        return
      }
      $tmp\_ohc \leftarrow F(tmp\_ohc)$
    }
  }
}
```

Figure 3. Algorithm for REQ message processing.

its encrypted data transmissions with its neighbors. Node s's cluster key KC_s is a key shared by s and all of s's verified neighbors. To set up KC_s, s generates a random number KC_s, and unicasts it to all its verified neighbor nodes, encrypted with their respective pair-wise keys. When a node s forwards a REQ message, it will encrypt the message with its cluster key KC_s.

3.3. Multiple Paths Set Up

Given the pair-wise and cluster keys, the process of setting up multiple routing paths is as follows:

1. Every node runs the echo-back process to identify its neighbor nodes and sets up pair-wise keys with its verified neighbor nodes. Then it sends its cluster key to each of its neighbor nodes encrypted using that neighbor's pair-wise key.
2. Each base station broadcasts its REQ message to its neighbor nodes.
3. When a sensor node receives the broadcast message, it processes the REQ message.

In step 2, the format of the REQ message is:

$$REQ : REQ||ID_s||E_{KC_s}(OHC||ID_B)$$

Where REQ is the type of the message, ID_s is the ID of the currently sending node s, ID_B is the ID of the base station who generated this REQ message, and OHC is that base station's one-way hash chain number.

When node x receives this REQ message, first it checks the sender ID. If s is x's verified neighbor, x decrypts the one-way hash chain number OHC with s's cluster key, then x uses the one-way function F and its cached OHC number of base station B to verify the new incoming OHC number. If the OHC is valid, x will replace its cached OHC number with this new incoming value, encrypt OHC with its own cluster key, and broadcast the newly encrypted REQ message. Figure 3 shows the algorithm for sensor node x to process the REQ message.

3.4. Maintaining node joins and leaves

If a node runs out of its battery or is damaged, it will leave the network. This dead node blocks the communication path of its child nodes. The redundant path approach can tolerate a certain number of such nodes leaving. In addition, the base stations will periodically collect network topology information to find the dead nodes, as described later. If a new node is added into the network, it can use the echo-back approach to find its verified neighbors, and can temporarily set one of its neighbor nodes as its parent node for data transmission purposes. When base stations flood new REQ messages, this node will then find its preferred parent node.

4. ANTI-TRAFFIC ANALYSIS COUNTERMEASURES

Data traffic in a sensor network is typically asymmetric. As sensor nodes report their data, the direction of the data movement is mostly towards the base station. This asymmetric communication pattern can aid an adversary in tracking down the location of a base station. This can result in the adversary launching serious attacks on the base station and eventually bringing down the entire sensor network. There are several ways to track the location of a base station:

1. If an adversary can understand the contents of a packet being transmitted, the adversary can correlate the packets that are forwarded towards the base station. This will allow the adversary to follow the direction of these packets towards the vicinity of the base station, leading to localized jamming and/or discovery and destruction of the base station.
2. If there is a time-correlation between when a node receives a packet and when it forwards that packet, an adversary can use this time correlation to find the direction towards the base station.
3. If there is no traffic control, a node that is near the base station will in general send data more frequently than the nodes that are farther away from the base station, because data accumulates as it is funneled towards the

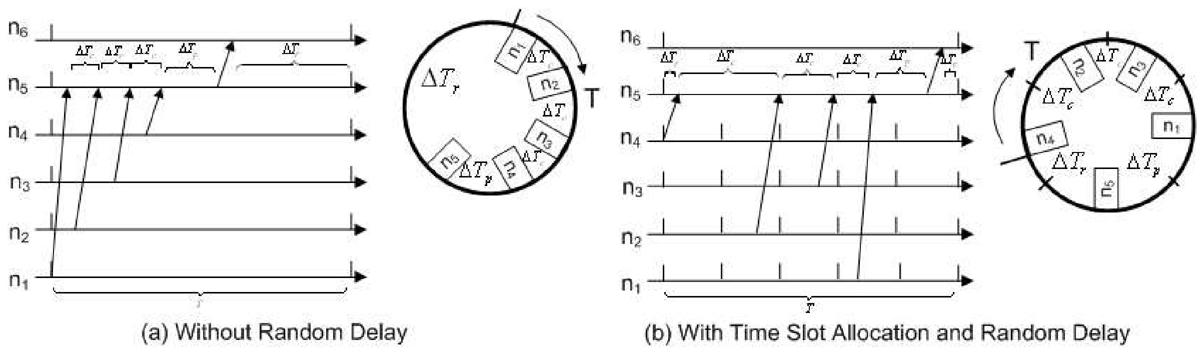


Figure 4. Decorrelating packet send times via random delays.

base station. By monitoring the data transmission rate, the adversary can track the location of the base station.

Different data transmission schemes may have different time-correlation patterns and different data sending rate constraints. In this paper, we propose anti-traffic analysis mechanisms to prevent an adversary from using any of these methods to discover the location of the base station under some common data transmission schemes. Note that it is difficult to track the location of the base station by monitoring REQ messages, because those messages occur infrequently and go far away from the base station. The goal of our anti-traffic analysis schemes is to prevent an adversary from finding the traffic directions by analyzing packet transmissions within its range. In particular, our goals are:

- An adversary cannot determine a packet destination by inspecting the contents of the packet.
- An adversary cannot find the data flow direction by analyzing the time correlation between the packets sent by child nodes and packets sent by their parent nodes.
- An adversary cannot find the data transmission direction by doing statistical analysis of the packet transmission rate of every node within its range.

For simplicity, in this section, we consider only those sensor networks that have one base station. All techniques proposed here can be extended to multiple base station networks as well.

4.1. Hidden Packet Destination Address

To hide the contents of a packet and its destination address, every node encrypts the destination address, packet type, and the contents of the packet with its cluster key. The current sender's address remains in plaintext so that the receiver can choose the correct cluster key to decrypt the packet. The format of a packet is

$$ID_{src}||E_{KC_{src}}(type||ID_{dst}||data)$$

When a node receives this packet, it checks ID_{src} and decides which cluster key to use to decrypt the packet. After decrypting the rest of the packet, a node checks if it is the destination of the packet.

The net effect is that the packet's entire appearance is transformed at every hop along its path, making it difficult for an eavesdropper to trace the path of the packet. Hop-by-hop reencryption spatially decorrelates the packet's appearance. Unless an attacker can compromise a sender's neighbor node and obtain the cluster key, it won't know the contents of the packet. If an attacker compromises a node s and obtains all the keys inside the node, it will be able to decrypt the packets sent by s's parent node, and can then track two hops towards the base station, but cannot track beyond that.

4.2. Decorrelating Packet Sending Times

Packet encryption can hide a packet destination, but cannot hide its sender. By carefully monitoring the packet sending time of every node, an adversary may get some information about data traffic flows. For example, if a parent node s receives a packet from its child node c and forwards that packet immediately, an adversary can observe the short time interval between s and c and eventually infer the parent-child hierarchy given sufficiently long observations.

To prevent this, we decorrelate the packet sending times between a parent node and its child nodes. Here we only consider the situation that every node sends data at the same rate. This situation occurs when every node regularly aggregates data from its children nodes and sends a result to its parent node. Suppose all child nodes and parent nodes report their data during time period T. Let's denote the time interval between two child nodes sending packets as Δt_c (we assume sensor nodes use a MAC layer protocol to avoid packet collisions), the time interval from the last child node sending data to the parent node sending data as Δt_p, and the time between a parent node sending data and its grandparent forwarding data as Δt_r. We denote $\overline{\Delta t_c}$, $\overline{\Delta t_p}$, $\overline{\Delta t_r}$ as the

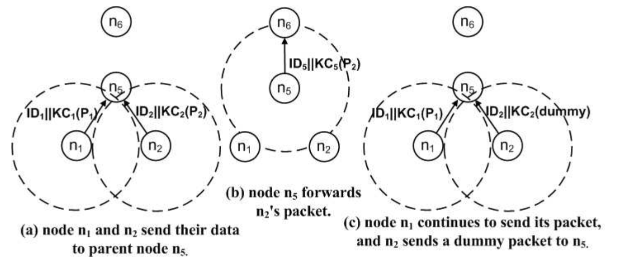

n6
n5
n1
n2
ID1||KC1(P1)
ID2||KC2(P2)
(a) node n1 and n2 send their data to parent node n5.
ID5||KC5(P2)
(b) node n5 forwards n2's packet.
ID2||KC2(dummy)
(c) node n1 continues to send its packet, and n2 sends a dummy packet to n5.

Figure 5. Rate control scheme.

average value of Δt_c, Δt_p, and Δt_r. If the differences between $\overline{\Delta t_c}$, $\overline{\Delta t_p}$ and $\overline{\Delta t_r}$ are observable, an adversary may be able to extract which node is the parent node after monitoring the network for an extended period of time.

If the parent node and child nodes send packets with the same rate, sensor nodes can introduce random delay between packet sending times. This makes the differences between $\overline{\Delta t_c}$, $\overline{\Delta t_p}$ and $\overline{\Delta t_r}$ unobservable. To do this, first the time period T is divided into m slots, if there are $m-1$ child nodes and 1 parent node. Every node is assigned a slot and randomly chooses a time within its slot to send its packet. For example, in Figure 4, the time slot assignment algorithm is centered at the parent node. The parent node informs each child node of its time slot with a secure unicast message. Nodes n_1 to n_4 are n_5's child nodes, and n_6 is n_5's parent node. Figure 4(a) shows every node sends its packet as soon as it can. The differences between Δt_c, Δt_p and Δt_r are correlated. Figure 4(b) shows that n_1 to n_5 occupy different time slots and each node sends its packet randomly within its time slot. The differences between Δt_c, Δt_p and Δt_r are indistinguishable. Experiments show that a sensor node only spends about 40 to 50 milliseconds to send a 36 bytes packet. Normally, a sensor reports data once a minute or tens of seconds. In a connected sensor network, a sensor node may have 10 to 20 neighbor nodes. So the time slot is big enough for a sensor node to successfully send its packet.

4.3. Controlling Packet Sending Rates

In the previous subsection, we assumed that every node sends packets at the same rate. However, in some cases, different sensor nodes may send packets with different rates. For example, the base station may require that each sensor node sends its neighborhood information (which contains the IDs of its identified neighbor nodes) back to the base station. We call this a *topology report*. The topology report helps a base station to update its complete network topology picture. The end user can use this information to learn what sensor nodes and base stations are unreachable. For the topology report messages, a parent node has to forward every message from its children nodes, and aggregation is avoided. If every node sends packet with the same rate, then nodes closer to the base station will sustain larger sending rates. By monitoring packet sending rates, an adversary can track the base station.

```
while (1) {
  send $P_s$ to parent node
  listen to packet sending of neighbor nodes
  if receive packet $p$
  if ($p.sender == parent\_node$) {
    if (($p == P_s$)||($p == dummy$)){
      $P_s \leftarrow dummy$
    }
  } else if ($p.sender \in s.children$) {
    if ($p \neq dummy \&\& P_s == dummy$) {
      $P_s \leftarrow p$
    }
  }
  wait for next time slot
}
```

Figure 6. Algorithm for packet sending control.

Our solution is to set the packet sending rate control between a parent node and its children nodes. That creates a uniform sending rate across the entire sensor network, so that every node behaves like every other node in terms of traffic volume. When a parent node has a packet in its buffer to send, it won't accept any packet from its children nodes. When the parent has sent out its packet, it accepts one packet from its children nodes and saves that packet into its buffer. All children nodes are monitoring the packet sent out by their parent node, because they have the parent's cluster key. If a child node finds that its packet was just transmitted by its parent node (that means its parent node has received its packet), or if it finds its parent begins to send dummy packets (that means the parent node has empty buffer), then the child begins to accept a new packet from its children nodes. Otherwise it will continue to send the same packet to its parent node. If a node doesn't have any packet to send, it just injects a dummy packet to its parent, until the whole topology reporting process stops. The base station can send a broadcast message to start and stop the topology reporting process. This rate control scheme is depicted in Figure 5, and figure 6 describes the algorithm. This algorithm implements rate control, and it robust in case the child node fails to hear the parent node forward its packet.

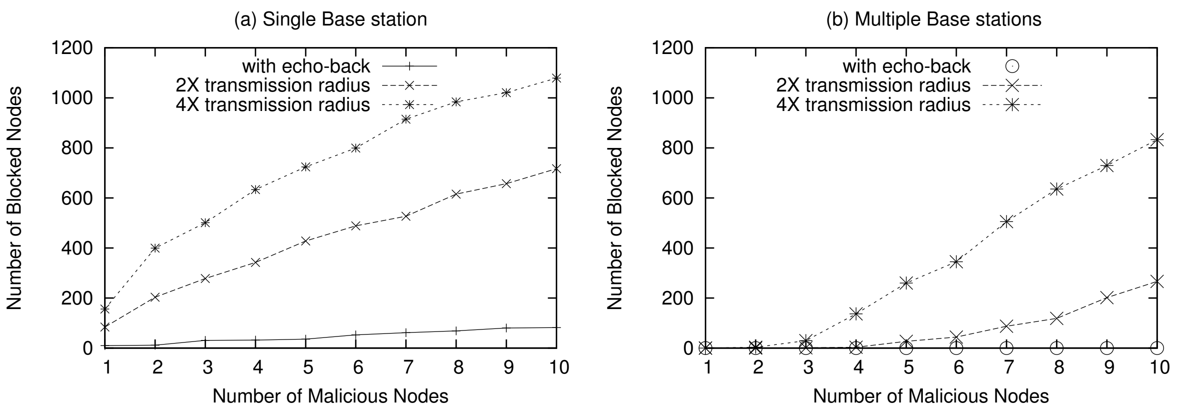


Figure 7. Effects of Rushing Attack During Multipath Routing Setup.

5. PERFORMANCE ANALYSIS

5.1. Overhead of Cryptographic Algorithms

A sensor node needs to save a global key, pair-wise keys, cluster keys, and one-way hash chain numbers. Suppose every key is 8 bytes. If a node has n neighbor nodes and there are k base stations, it uses $8 \times (2n + k + 2)$ bytes to save all keys. For example, if there are 4 base station, and a node has 10 neighbor nodes, it uses 208 bytes for all keys. If the keys are not changed very often, e.g. global key and pair-wise keys, they can be saved in the 128KB flash memory or the 4KB embedded EEPROM.

To evaluate the performance of computing overhead of cryptographic algorithms in REQ flooding and destination address encryption, we implemented encryption/decryption algorithms, and one-way hash chain verification on Berkeley MICA1 sensor motes [1]. We chose RC5 (with 12 rounds) as the block cipher to implement these algorithms. Table 1 shows performance of our implementation. The experiment shows that the overhead of verifying the one-way hash chain number on sensor nodes is not prohibitive.

5.2. Performance of secure multipath set up

To evaluate effectiveness of multipath to multiple base station routing, we simulated our routing path set up scheme and measured the number of nodes blocked by adversary. We simulated the case that there is only one base station at the center of a network, and the case that there are 4 base stations at 4 corners of a network. We simulated the case that the malicious node can have from 2 to 4 times data transmission radius that a normal node has when we didn't apply echo-back approach, and the case that the malicious node's effective transmission range is as same as a normal node's data transmission range when we applied echo-back approach. To measure the number of blocked nodes, we randomly distributed 2000 nodes in a network area with density that every node has about 16 neighbors in average. We randomly selected the malicious nodes from 1 to 10 among the 2000 nodes, and simulated how many nodes will be blocked by malicious nodes if malicious nodes launch rushing attack with 1, 2, and 4 times of normal data transmission range respectively. We repeated the test for 100 times. Every time we radnomly distributed the 2000 nodes and randomly selected malicious nodes. Figure 7 shows the average number of nodes blocked by malicious nodes.

Figure 7 (a) shows the results for the single base station case. We can see that the echo-back approach is very effective in preventing the rushing attack. For example, if adversaries launch rushing attacks at 10 different places and their packets can reach 4 times away further than a normal node does, they can block about half nodes in the network. In comparison, when echo-back is used to defend against rushing attacks, adversaries can only block about 5% of the nodes in the network. Figure 7 (b) shows the results for the multiple base station case. From this figure, we can see again that the echo-back approach is still very effective against rushing attacks. In addition, compared with figure 7 (a), we can see that multiple path routing to multiple base stations provides considerably more robust network connectivity than the single base station approach, especialy in combination with the echo-back defense.

	Speed (msec)	Code (Bytes)	Data (Bytes)
Encryption (30bytes)	1.94	1488	112
Decryption (30bytes)	2.02	1518	112
One-way hash chain	4.18	1768	136

Table 1. Overhead of Cryptographic Algorithms

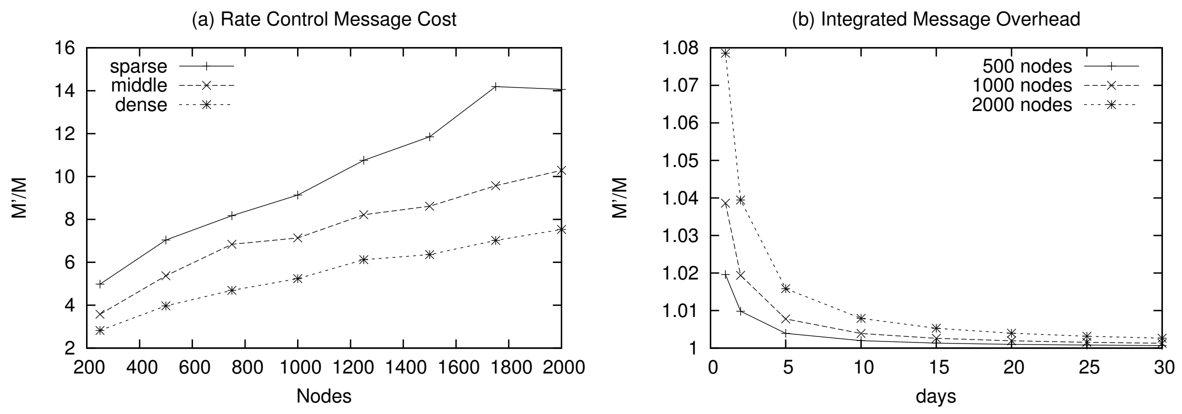


Figure 8. Overhead of Anti-traffic Analysis.

5.3. Anti-traffic Analysis Message Overhead

We define $C = \frac{M'}{M}$ to measure the data transmission overhead of our anti-traffic analysis strategy, where C is the cost measurement, M is the number of messages without the anti-traffic analysis strategy, and M' is the number of messages with the anti-traffic analysis strategy. In our experiments, we simulated and measured the message overhead of the rate control scheme, since it introduces extra "dummy" packets. We ran three groups of tests. For each group of tests, we employed a different network topology. These networks differed from one another in the number of nodes, but had the same node density. The number of nodes varied from 250 to 2000. For each test, sensor nodes were randomly deployed in the network area. We set up routing, and measured M' and M. For the same number of nodes with the same network density, we repeated the test 50 times and calculated the average value. Figure 8 (a) shows the simulation result of $\frac{M'}{M}$ for three different network densities. We can see that the overhead of our rate control strategy increases as the size of the network increases (Our initial analysis and experiments show that $\frac{M'}{M} \propto \sqrt{N}$, where N is the size (number of nodes) of network). That means the rate control overhead is not scalable corresponding to the size of network.

However, if topology information is not required frequently, the overhead of the rate control scheme only occupies a small part of the total cost. The network traffic is dominated by regular sensed data report, whose anti-traffic message overhead is 1. Figure 8 (b) shows the total message overhead combining sensor data packets and topology reports over an intermediate density network. We assume that every node reports its data once per minute, and the base station requires a topology report ever 1 day to 30 days. Figure 8 (b) shows that the total overhead reduces as the base station requires topology reports less frequently. For example, if the topology report is performed once a week, total overhead is less than 1.01. In this context, the overhead of sending "dummy" packets is much less noticeable.

6. Related Work

Sensor network security is a critical issue in sensor network research [22], [19], [15]. A. Perrig et. al [19] addressed secure communication in resource-constrained sensor networks, introducing two low-level secure building blocks, SNEP and μTESLA. A. Wood and J. Stankovic [22] provided a survey of many kinds of denial of service attacks in sensor networks and discussed defense technologies. C. Karlof and D. Wagner [15] analyzed security flaws of various routing protocols on WSNs, and proposed countermeasures to enhance sensor network routing. To defend against the rushing attack, this paper proposed that every node only process beacon messages through bidirectional links as well as verified neighbor nodes. However, the paper uses a trusted base station for neighborhood verification, which is not scalable for a large sensor network. We propose to use a global key or random pre-destributed keys for neighbor node verification.

Anti-traffic analysis is a interesting topic in network privacy. The onion routing protocol disguises who talks to whom on the Internet by layered encryption and by forwarding received messages in a random order [12]. However, the onion protocol works for an arbitrary pair of communication nodes, which is common in Internet but not in sensor network. In addition, the onion router stores a large number of messages before forwarding them in a different order. A sensor node doesn't have enough memory to store lots of packets.

While the issue of intrusion tolerance has been known for quite some time [11][4], recent increase in the need for safety-critical systems has significantly raised research activity in this area. Recent projects addressing intrusion tolerance include [5][20][21][23]. All these projects are aimed at providing intrusion tolerance capabilities in a traditional, resource-rich computing environment.

INSENS [8] proposed an intrusion tolerant protocol that set up multiple paths in a WSN. However, in INSENS, every sensor node needs to send a "feedback" message to the base station, which is inefficient and not scalable. In addition, REQ message is vulnerable to rushing attacks. Our mutltipath to multiple base stations routing scheme addressed these security flaws.

7. Conclusion and Future Work

In this paper, we have addressed important security and intrusion-tolerance issues in building a distributed wireless sensor network. Two intrusion tolerance schemes are proposed to defend a WSN against attacks focused on isolating base stations and tracking base stations. First, the secure set up of multiple paths to multiple base stations is introduced to tolerate isolation of a base station. Mechanisms like one-way hash chains and the echo-back algorithm are proposed to prevent spoofing, DOS attacks, and rushing attacks. Second, anti-traffic analysis strategies like hop-by-hop cluster key encryption/decryption and sending rate control are offered to disguise the location of base stations from eavesdroppers. Future work includes designing anti-traffic schemes for other data transmission schemes, such as general data aggregation reporting, and providing low overhead rate control mechanisms.

8. Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

[1] Tinyos website. http://webs.cs.berkeley.edu/tos/.

[2] H. Abrach, S. Bhatti, J. Carlson, H. Dui, J. Rose, A. Sheth, B. Shucker, J. Deng, and R. Han. Mantis: System support for multimodal networks of in-situ sensors. In *(WSNA'03)*, San Diego, CA, USA, September 2003.

[3] U. A. F. ARGUS Advanced Remote Ground Unattended Sensor Systems, Department of Defense. Argus. http://www.globalsecurity.org/intell/systems/arguss.htm.

[4] L. Blain and Y. Deswarte. An intrusion tolerant security server for an open distributed system. In *1st European Symposium in Computer Security*, Toulouse, France 1990.

[5] C. Cachin and J. A. Poritz. Secure intrusion-tolerant replication on the internet. In *2002 IEEE International Conference on Dependable Systems and Networks (DSN'02)*, Washington D.C, USA, June 2002.

[6] J. Carlson, R. Han, and et.al. Rapid prototyping of mobile input devices using wireless sensor nodes. In *WMCSA'03*, Monterey, California, USA, October 2003.

[7] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, May 2003.

[8] J. Deng, R. Han, and S. Mishra. The performance evaluation of intrusion-tolerant routing in wireless sensor networks. In *IPSN'03*, Palo Alto, CA, USA, April 2003.

[9] W. Du, J. Deng, Y. Han, and P. Varshney. A pairwise key predistribution scheme for wireless sensor networks. In *10th ACM Conference on Computer and Communications Security (CCS'03)*, Washington D.C, USA, October 2003.

[10] L. Eschenauer and V. Gigor. A key-management scheme for distributed sensor networks. In *Conference on Computer and Communications Security, (CCS'02)*, Washington DC, USA, November 2002.

[11] J.-M. Fray, Y. Deswarte, and D. Powell. Intrusion-tolerance using fine-grain fragmentation-scattering. In *1986 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, April 1986.

[12] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. *Communications of ACM*, 42(2), February 1999.

[13] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Cullar, and K. Pister. System architecture directions for network sensors. In *Nineth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'00)*, Cambridge, MA, USA, November 2000.

[14] Y. Hu, A. Perrig, and D. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *2nd ACM Workshop on Wireless Security (WiSe'03)*, San Diego, CA, USA, September 2003.

[15] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad Hoc Networks*, 1(2-3), September 2003.

[16] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *CCS'03*, Washingon D.C, USA, October 2003.

[17] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *5th Symposium on operating systems design and implementation (OSDI'02)*, Boston, MA, December 2002.

[18] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA'02*, 2002.

[19] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar. Spins: Security protocols for sensor networks. *Wireless Networks Journal(WINET)*, 8(5):521–534, September 2002.

[20] H. V. Ramasamy, P. Pandey, J. Lyons, M. Cukier, and W. H. Sanders. Quantifying the cost of providing intrusion tolerance in group communication systems. In *DSN'02*, Washington D.C, USA, June 2002.

[21] D. Sames, B. Matt, B. Niebuhr, G. Tally, B. Whitmore, and D. Bakken. Developing a heterogeneous intrusion tolerant corba system. In *DSN'02*, Washington D.C, USA, June 2002.

[22] A. Wood and J. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, October 2002.

[23] T. J. Wu, M. MalKin, and D. Boneh. Building intrusion tolerant applications. In *8th USENIX Security Symposium*, pages 79–91, Washington D.C, USA, August 1999.