Chapter 1

SWAT: A Decentralized Self-healing Mechanism for Wormhole Attacks in Wireless Sensor Networks

Chonho Lee and Junichi Suzuki

Department of Computer Science
University of Massachusetts, Boston
{*chonho and jxs*} *@ cs.umb.edu*

This paper proposes and evaluates a decentralized self-healing mechanism that detects and recovers from wormhole attacks in wireless multi-hop sensor networks. Upon detecting a wormhole attack, the proposed mechanism, called SWAT, identifies the locations of malicious nodes (or wormhole nodes), isolates them from the network and recovers the routing structure distorted by them. SWAT is the first mechanism that performs both wormhole node isolation and routing structure recovery against wormhole attacks. Unlike many other wormhole detection mechanisms, SWAT does not require any extra networking facilities (e.g., timing analysis and localization facilities) as well as special hardware (e.g., GPS). Instead, it uses network connectivity information only in a decentralized manner. Simulation results show that SWAT yields 100% wormhole attack detection, 0% false detection, 100% wormhole node isolation and 0% false isolation in dense networks. The results also show that SWAT outperforms multi-path routing mechanisms in terms of control overhead and power consumption and outperforms another connectivity-based detection mechanism in terms of false isolation rate and recovery efficiency.

1.1. Introduction

The wormhole attack is one of the most severe security threats in wireless multi-hop sensor networks.[1–3] In this attack, two malicious nodes, called *wormhole nodes*, are strategically placed at distant points in the network, and they are connected in one hop via wireless out-of-band link or wired link called a *wormhole link*. Through this link, wormhole nodes make far apart nodes believe they are close with each other. Taking advantage of this deceptive belief, one wormhole node attracts data packets from its neighboring nodes (called *affected* nodes), tunnels them through a wormhole link and replays them at the other wormhole node.

A major impact of wormhole attacks is that wormhole nodes collude to distort routing structure and take a full control of the data traffic through a wormhole link. Wormhole attacks can disrupt any mechanisms that rely on topological proximity. For example, connectivity-based localization mechanisms fail such that far apart

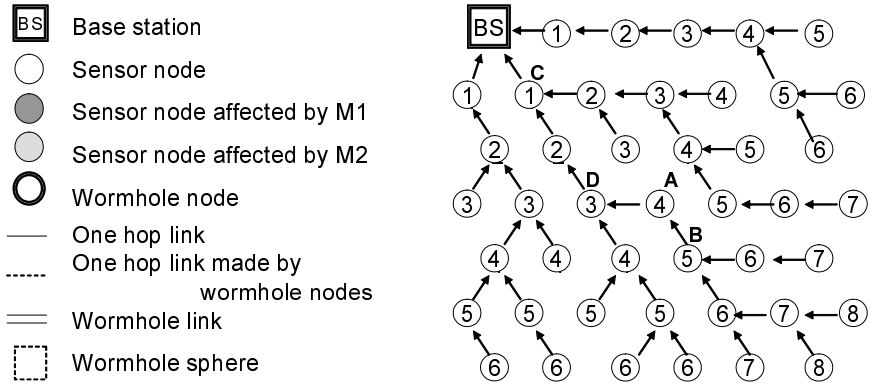
nodes are localized nearby. This can result in a critical failure in wireless sensor networks because location information is always important in many protocols and applications. Wormhole nodes can also degrade the latency of data delivery from individual nodes to the base station and increase power consumption of nodes by imposing extra node-to-node data transmissions when one wormhole node attracts packets near the base station and replays them at the other that is far from the base station.

Another major impact of wormhole attacks is that wormhole nodes can easily perform secondary (passive and active) attacks anytime. For example, wormhole nodes may eavesdrop, delay, modify or selectively drop incoming packets. They may also impose extra routing structure reconstruction, thereby increasing power consumption of nodes, by turning off a wormhole link periodically.

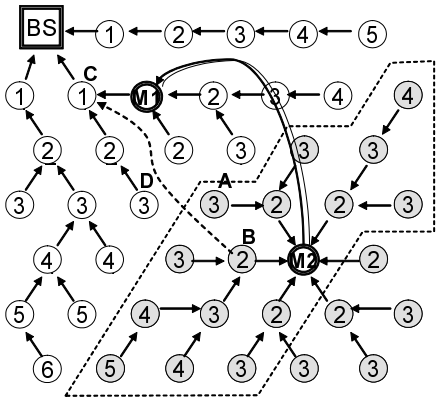
A challenging issue of wormhole attacks is that attackers do not need to know the cryptographic measures used in the network and posses cryptographic keys. Wormhole attacks cannot be defeated even if the network retains confidentiality and authenticity because wormhole nodes do not create extra packets but simply replay packets that already exist in the network. As far as wormhole nodes are passive, they are invisible.

This paper proposes and evaluates a decentralized self-healing mechanism that detects and recovers from wormhole attacks. Upon detecting a wormhole attack, the proposed mechanism, called SWAT[a], isolates wormhole nodes from the network by eliminating links connected to them, and recovers the routing structure distorted by the wormhole nodes. SWAT is the first mechanism that performs both wormhole node isolation and routing structure recovery against wormhole attacks. With scalability in mind, SWAT is designed as a decentralized in-network detection mechanism that uses network connectivity information only. Simulation results show that SWAT yields 100% wormhole attack detection, 0% false detection, 100% wormhole node isolation and 0% false isolation in dense networks. The results also show that SWAT outperforms multi-path routing mechanisms in terms of control overhead and power consumption and outperforms another connectivity-based detection mechanism in terms of false isolation rate and recovery efficiency.

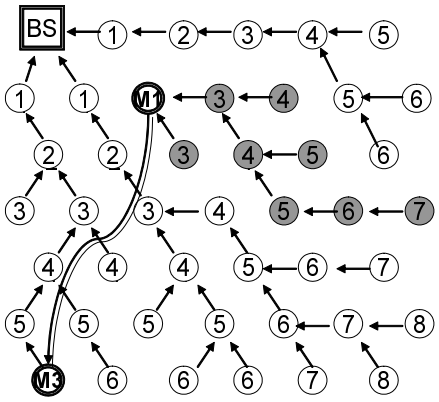
This paper is organized as follows. Section 1.2 overviews wormhole attacks. Section 1.3 compares SWAT with existing work. Section 1.4 describes the self-healing process and design rationales of SWAT. Section 1.5 shows a series of simulation results to obtain the performance implications of SWAT and compare SWAT with other wormhole detection mechanisms. Section 1.6 concludes this paper with some discussion on future work.



(a) Normal, Undistorted Routing Structure



(b) Routing Structure Distorted by M1 and M2 (wormhole nodes)



(c) Routing Structure Distorted by M1 and M3 (wormhole nodes)

Fig. 1.1. Example Wormhole Attacks

1.2. Preliminaries: Wormhole Attacks

The wormhole attack was first identified in wireless ad-hoc networks by Hu et al.,[4] and it was first discussed in the context of wireless sensor networks by Karlof et al.[5] Figure 1.1 shows example wormhole attacks. Figure 1.1 (a) shows a normal, undistorted tree-based routing structure. Figures 1.1 (b) and (c) show two routing structures distorted under wormhole attacks. A number shown in each node (circle) indicates the topological distance (hop count) to the base station.

In Figure 1.1 (b), a wormhole node, M2, attracts packets from its neighboring (shaded) nodes and forwards them to the other wormhole node, M1. Using a wormhole link, M1 and M2 make their neighboring nodes believe that they are close. For example, the node B believes that the node C is a direct neighbor. As a result, B forwards packets to the base station through M2, rather than A, because the rout-

[a]Self-healing mechanism against Wormhole ATtacks

ing path through M2 is the shortest toward the base station. This way, M2 distorts its surrounding routing structure. A wormhole attack in Figure 1.1 (b) forms a *wormhole sphere*, which is a set of affected nodes that change their routing paths due to a wormhole attack. For example, in Figure 1.1 (a), the node B points the node A as its parent node in its routing path; however, due to a wormhole attack, B changes its parent node from A to M2 in Figure 1.1 (b). Thus, B is included in a wormhole sphere. When a wormhole is formed, the network is topologically separated. For example, the node A is no longer connected to the node D due to a wormhole attack. (See Figures 1.1 (a) and (b).) This link loss contributes to a network separation. A set of nodes that separate the network at the edge of a wormhole sphere is called *edge nodes*. The node A is an edge node of a wormhole sphere.

In Figure 1.1 (c), a wormhole node, M1, receives packets from eight (shaded) nodes in its subtree and forwards them to the other wormhole node, M3. M1 distorts routing structure by diverting incoming packets. This packet diversion incurs a higher number of node-to-node packet transmissions toward the base station, thereby imposing higher power consumption of nodes.

1.3. Related Work

Hu et al. propose the notions of geographical and temporal leashes.[6] A geographical leash ensures that a packet receiver exists within a certain range (e.g., transmission range) from a packet sender. When a node receives an out-of-range packet, it eliminates the packet under the assumption that the packet arrived through a wormhole link. This detection method requires each node to know its precise location. Several variations of geographical leashes have been proposed by Capkun et al.,[7] Wang et al.,[8] Hu et al.[9] and Sastry et al.[10] These mechanisms require each node to equip an extra device such as a GPS receiver, directional antenna or ultrasound transceiver in order to obtain its location information.

A temporal leash ensures that a packet receiver receives packets from their senders within a certain time bound. When a node receives an out-of-date packet, it eliminates the packet under the assumption that the packet arrived through a wormhole link. This detection method requires nodes to perform precise clock synchronization and timing analysis on the order of hundred nanoseconds to a few milliseconds.[11,12] Zhen et al. propose a similar approach that measures packet round-trip time between a node and its neighbors in order to examine whether each neighbor is a physical neighbor or not.[13] Unlike the detection methods based on geographical and temporal leashes, SWAT does not require any extra networking facilities (e.g., time synchronization, timing analysis and localization facilities) as well as special hardware devices such as GPS receivers. This can contribute to reduce the costs to deploy large-scale WSNs, which deploy a large number of nodes.

Poovendran et al.,[14] Lazos et al.[15] and Khalil et al.[16] propose device-free de-

tection mechanisms. They assume special nodes, such as anchor, locator and guard nodes, which need to be always uncompromised. The special nodes know their locations and localize their neighboring nodes. Using the location information, individual or special nodes detect and eliminate the packets transmitted via wormhole links. In contrast, SWAT does not require any special nodes and does not rely on location information. Instead, it uses network connectivity information only.

Phuong et al. propose a detection method that leverages network traffic information.[17] Each node monitors the number of incoming/outgoing packets, and detects a wormhole attack by identifying an imbalance or a dramatic change of traffic volume based on a CUSUM (Cumulative Sum) algorithm. In this method, each node is required to utilize a large amount of memory space to keep traffic record for a certain time period. In contrast, SWAT uses network connectivity information and requires less memory space.

Vajda et al.[18] propose a detection method using network connectivity information. The base station periodically examines the number of direct neighbors of each node and statistically determines a wormhole attack under the assumption that nodes have a higher number of neighboring nodes around wormhole nodes. This centralized method does not scale well in large-scale WSNs. The more often the base station collects connectivity information from individual nodes, the higher power consumption they need to incur. The less often the base station collects connectivity information, the longer delay it detects wormhole attacks. Unlike this method, SWAT is designed as a decentralized in-network detection method.

Hou et al. discuss a decentralized detection method using connectivity information.[19] They claim that every node should always have an alternative path to each of its direct neighbor nodes in addition to a direct (one-hop) path. Each node seeks such alternative paths to its direct neighbors and alerts if they are not found. The control overhead of this method is quite expensive because each node keeps flooding control packets. In SWAT, each node does not always keep flooding control packets, thereby retaining lower control overhead.

Wang et al.[20] and Xu et al.[21] propose other detection methods using connectivity information. Both methods obtain the topological distance (hop count) from each node to the base station, and construct a distance map for all pairs of nodes by using multi-dimensional scaling techniques. Based on the distance map, the base station[20] or each node[21] detects a wormhole attack by identifying the routing structure distorted by wormhole nodes. These methods are computationally expensive; its computational complexity is $O(N^3)$ where N is the number of direct neighbors of each node. In contrast, SWAT maintains lower computational complexity ($O(N^2)$).

All of the aforementioned methods can detect wormhole attacks; however, they cannot identify the locations of wormhole nodes and neutralize them by isolating them from the network. In contrast, SWAT inherently considers and performs wormhole node isolation by eliminating the links connected to them so that no more packets will be transmitted via wormhole link in the future.

Maheshwari et al. propose a decentralized detection method that is most similar to SWAT.[22] This method leverages connectivity information to isolate wormhole nodes from the network. Although it is more computationally efficient than any other decentralized detection methods including SWAT, it cannot pinpoint the locations of wormhole nodes. As a result, it isolates several normal, uncompromised nodes around wormhole nodes in order to guarantee isolating all wormhole nodes. Unlike this method, SWAT is designed to minimize false isolation. In addition, it recovers the routing structure distorted by wormhole nodes. Maheshwari et al. do not consider routing structure recovery.

1.4. SWAT: The Proposed Self-healing Mechanism

SWAT performs its self-healing process in two phases: *detection* and *recovery* phases. In the detection phase, SWAT detects a wormhole attack and identifies the locations of wormhole nodes. In the recovery phase, SWAT isolates wormhole nodes from the network and recovers distorted routing structure.

SWAT operates under a few assumptions. First, sensor nodes are uniformly and densely deployed in an observation area, and each node has the same transmission range. Second, the base station constructs a routing tree structure by propagating a control packet to individual nodes on a hop-by-hop basis. (Figure 1.1 (a) shows an example routing tree.) Based on the routing tree, each node forwards incoming data packets to its next hop (parent) node. Third, each node maintains a neighbor list that contains the connectivity information on physical neighbor nodes in one and two hops away. For each neighbor node, the list maintains its node ID, its parent node ID and the hop count to the base station. In order to obtain and update the neighbor information, each node exchanges its direct neighbor's information with the direct neighbors.

1.4.1. *Detection Phase*

In the detection phase, SWAT detects a wormhole attack and identifies the locations of wormhole nodes. Using its neighbor list, each node monitors the network connectivity with physical neighbor nodes that are one and two hops away. When a node finds any connectivity anomaly with them, it produces a control packet, called *danger signal*. A danger signal indicates the existence of a wormhole attack. Danger signals are used to locate wormhole nodes and trigger the next phase, recovery phase, which isolates wormhole nodes and recovers distorted routing structure. There are three types of danger signals: *DS1*, *DS2* and *DS3*. Each of DS1 and DS2 signals indicates that a node is connected to a wormhole node. A DS3 indicates that a node exists at the edge of a wormhole sphere.

A DS1 indicates a connectivity anomaly between a normal (uncompromised) node and a wormhole node. Using its neighbor list, every node examines how many nodes each of its physical direct neighbors is connected to. Given an assumption

that nodes are uniformly deployed in an observation area, each node should maintain the same number of physical direct neighbors. However, when a node is connected to a wormhole node, it is likely the node has a higher number of physical direct neighbors. For example, in Figure 1.2, wormhole nodes M1 and M2 make the node A have 11 direct neighbors, although it should have six direct neighbors as B does. Therefore, a node produces a DS1 when it finds a neighboring node who has a less number of direct neighbors than it does. When a DS1 is produced on a node, it stays on the node.

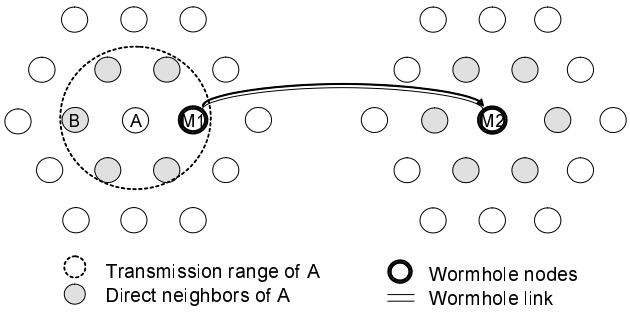
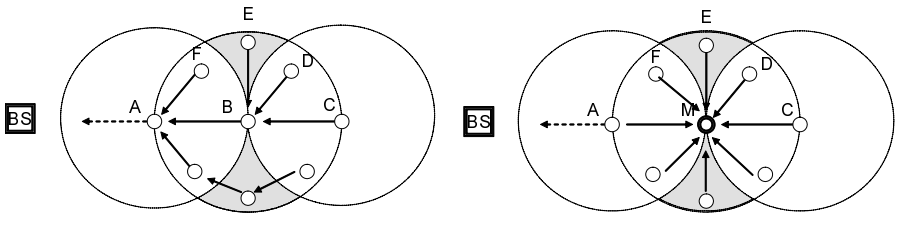


Fig. 1.2. An example case where a DS1 is produced

Similar to DS1, a DS2 also indicates a connectivity anomaly between a normal (uncompromised) node and a wormhole node. Using its neighbor list, each node counts the number of two-hops-away physical neighbors whose parent node is same as its own. It is likely the number grows under a wormhole attack. For example, Figure 1.3(a) shows a normal, undistorted routing tree. In this example, the node C shares its parent node (B) with only one two-hops-away node (E). However, under a wormhole attack (Figure 1.3(b)), five two-hops-away nodes, including E, F and A, point the node M as its parent as C does. Therefore, when the number of two-hops-away neighboring nodes that point the same parent node exceeds a threshold Th_2, a node produces a DS2. The produced DS2 stays on the node.

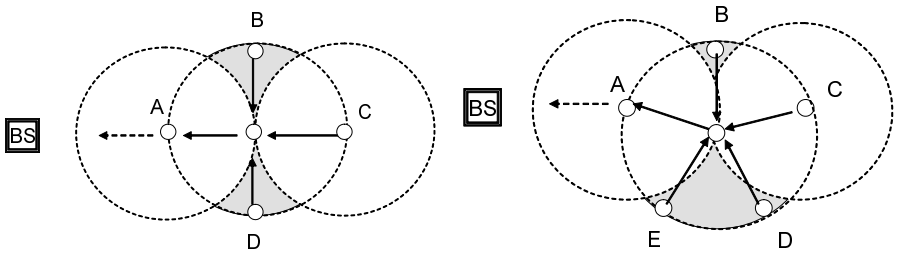


(a) No wormhole attack (b) Under a wormhole attack

Fig. 1.3. An example case where a DS2 is produced

The DS2 production depends on the value of Th_2 and network density, which

is measured as the number of physical direct neighbor nodes of each node. If both values are low, a node may produce a false positive DS2. In Figure 1.3(a), the node C can share its parent with the nodes in shaded areas (e.g., the node E) and may produce a false positive DS2. Therefore, Th_2 needs to be configured so that it can minimize false positive DS2 production. For example, when the number of physical direct neighbors of each node is four (Figure 1.4(a)), there can exist up to two two-hops-away nodes (B and D) that share the parent with the node C. If there are more than two such nodes, it is likely the node C is connected with a wormhole node. Therefore, the value of 2 is used for Th_2 when the number of direct neighbors of each node is four. When the number of physical direct neighbors of each node is five, there can exist up to three two-hops-away nodes (B, D and E) that share the parent with the node C (Figure 1.4(b)). If there are more than three such nodes, it is likely the node C is connected with a wormhole node. Therefore, the value of 3 is used for Th_2 when the number of physical direct neighbors of each node is five.



(a) The number of neighboring nodes = 4 (b) The number of neighboring nodes = 5

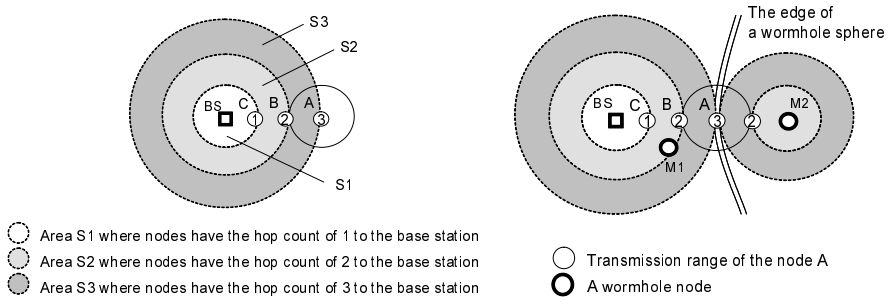
Fig. 1.4. Two-hops-away neighboring nodes in different degrees of network density

A DS3 indicates that a node exists at the edge of a wormhole sphere. Using a neighbor list, each node examines the following condition, and if it is satisfied, the node produces a DS3.

$$\frac{n_{hc}}{N} > Th_3 \tag{1.1}$$

n_{hc} denotes the number of physical direct neighbor nodes that have the same hop count to the base station. N denotes the total number of physical direct neighbor nodes. Without wormhole attacks (Figure 1.5(a)), each node has physical direct neighbors whose hop counts are in [hc-1, hc+1], where hc is its own hop count to the base station. For example, the node B is two hops away to the base station, and A and C (B's direct neighbors) are one and three hops away, respectively. In contrast, when a wormhole sphere is formed by a wormhole attack, it is likely n_{hc} grows around the wormhole sphere's edge. For example, in Figure 1.5(b), the wormhole node M2 attracts packets from its neighboring nodes and forwards them to M1; the edge of a wormhole sphere exists around the node A. In this case, the number of nodes that are three hops away to the base station becomes larger within

A's transmission range. (Note that the area S3's size is larger in Figure 1.5(b) than Figure 1.5(a).) Therefore, when $\frac{n_{hc}}{N}$ exceeds a threshold, Th_3, on a node, the node produces a DS3. When a DS3 is produced on a node, it is sent to the nodes between the node and its hc-hops-away parent node. The propagated DS3 stays on each node between its originator node to its hc-hops-away parent node.



(a) No wormhole attack (b) Under a wormhole attack

Fig. 1.5. An example case where a DS3 is produced.

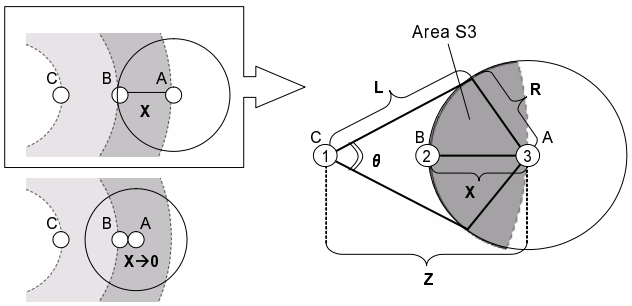


Fig. 1.6. Size of an area where nodes have the same hop counts to the base station

A proper value of Th_3 is determined by analyzing the maximum size of an area where nodes have the same hop counts to the base station. Figure 1.6 focuses on the S3 area around the node A in Figure 1.5(a). The size of $S3$ is maximized when $x = 0$; i.e., $Z = R$ where R is A's transmission range. It is approximately $1.913R^2$ as derived below.

$$\begin{aligned} Area(S) &= \int_R^{2R} ArcLength(L, \theta)\, dL \\ &= \int_R^{2R} L \cdot \theta\, dL \\ &= \int_R^{2R} 2L \cos^{-1}\left(\frac{L^2 + Z^2 - R^2}{2LZ}\right) dL \end{aligned}$$

$$= \int_{R}^{2R} 2L\cos^{-1}(\frac{L}{2R})\, dL \approx 1.913\ R^2$$

Given the maximum size of $S3$ under no wormhole attacks, $Area(S)/\pi R^2 \approx 0.6$ is used for Th_3.

1.4.2. *Recovery Phase*

In the recovery phase, SWAT isolates wormhole nodes from the network and recovers distorted routing structure. This recovery process is executed with three types of control packets, called *recovery packets*: *RCV1*, *RCV2*, and *RCV3*. RCV1 and RCV2 are designed to isolate wormhole nodes from the network by removing links connected to them. RCV3 is designed to recover the routing structures distorted in wormhole spheres.

Each recovery packet is produced on a node when a certain combination of danger signals (i.e., DS1, DS2 and DS3) is available on the node. In other words, danger signals trigger the production of recovery packets. Of eight possible combinations of three types of danger signals, SWAT considers four combinations to produce recovery packets. (See Table 1.1.) Each combination (or each node state) is represented as a sequence of three binary numbers. The value of 1 means that a corresponding danger signal is available, and the value of 0 means that it is not available. For example, (1, 1, 1) denotes that all of DS1 and DS2 and DS3 signals are available on a node. In this case, a RCV2 and a RCV3 are produced. In (1, 0, 1), (0, 1, 0) and (0, 0, 1), no recovery packets are produced in order to minimize the chances to trigger the recovery phase with potential false positive danger signals.

Table 1.1. Production of recovery packets

The state of a node (DS1, DS2, DS3)	Produced recovery packet(s)
(1, 1, 0)	RCV1
(1, 0, 0)	RCV1
(1, 1, 1)	RCV2 and RCV3
(0, 1, 1)	RCV2 and RCV3
(1, 0, 1)	None
(0, 1, 0)	None
(0, 0, 1)	None
(0, 0, 0)	None

An RCV1 isolates a wormhole node from the network. An RCV1 is produced when a node is in the (1, 1, 0) or (1, 0, 0) state (Table 1.1). Once an RCV1 is produced on a node, it is broadcasted to the node's physical direct neighbors. When a node receives an RCV1, it removes the RCV1's sender node from its neighbor list. If the RCV1 sender is its parent node, it selects a new parent node from its neighbor list. The new parent node must not be the RCV1 receiver's child node. Figure 1.7

shows an example of the recovery phase with RCV1 packets. In this example, the node B believes that its parent is the node F due to the wormhole link between M1 and M2. When B receives an RCV1 packet(s) from a shaded node(s) out of its physical direct neighbors (e.g., D, F, G and I), it removes those nodes from its neighbor list. (Note that those shaded nodes can be in the (1, 1, 0) or (1, 0, 0) state.) B selects the new parent from its neighbor list; in this example, B selects E. This way, B avoids to connect with a wormhole node (M1) so that it will never send packets to M1 in the future. Similarly, in Figure 1.7, the node G believes its parent is the node A due to a wormhole link. Upon receiving an RCV1 packet(s) from some of its physical direct neighbors including I, F, A, B and D, G removes those nodes from its neighbor list and selects another node (J) as the new parent. Through this process, G detours incoming packets to J rather than sending them to a wormhole node (M2). In Figure 1.7, wormhole nodes (M1 and M2) are isolated when both A and G perform the recovery phase with RCV1 packets.

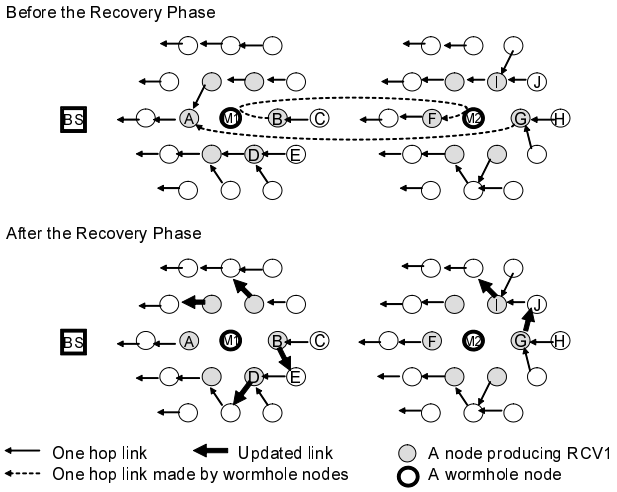


Fig. 1.7. An example of the recovery phase with RCV1 packets

Similar to an RCV1, an RCV2 isolates a wormhole node from the network. It is specifically designed to isolate a wormhole node that forms a wormhole sphere (e.g., M2 in Figure 1.1(b)). It is produced when a node is in the (1, 1, 1) or (0, 1, 1) state. Once an RCV2 is produced on a node, it is broadcasted with the hops-to-live count of two toward the nodes that are physically two hops away. When a node receives an RCV2, it removes a link to its parent node and selects the RCV2's sender node as the new parent. Figure 1.8 shows an example of the recovery phase with RCV2 packets. In this example, a wormhole node, M, attracts packets from its physical direct neighbors, and the node F broadcasts an RCV2 packet. Upon receiving the RCV2 packet, the nodes A and E change their parent from M to F.

They broadcast the RCV2 again to D and G; then D and G changes their parents to E and A, respectively. The wormhole node M is isolated when all of its physical direct neighbors perform the recovery phase with RCV2 packets.

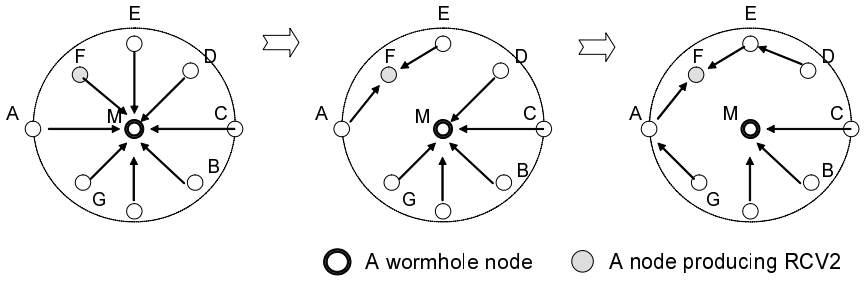


Fig. 1.8. An example of the recovery phase with RCV2 packets

An RCV3 recovers the routing structure distorted in a wormhole sphere. It is produced when a node is in the (1, 1, 1) or (0, 1, 1) state. Once a node produces an RCV3, it sends the RCV3 to its child nodes and changes its parent node to one of the child nodes. If a node does not have any child node when receiving an RCV3, the node discards the RCV3 and selects one of its physical direct neighbors as its parent node. Figure 1.9 shows an example of the recovery phase with RCV3 packets. In this example, a wormhole node, M, attracts packets from its neighboring nodes, and the node A produces an RCV3. A sends the RCV3 to B and C, and change its parent from M to B. When B receives the RCV3 from A, it forwards it to E and D, and it changes its parent from A to E. When C receives the RCV3 from A, it forwards it to F and G, and it changes its parent from A to G. Since F and G exist on the edge of a wormhole, they do not have child nodes. Therefore, they discard incoming RCV3 packets and selects one of its physical direct neighbors as its parent node. The wormhole node M is isolated when all of its neighboring nodes perform the recovery phase with RCV3 packets.

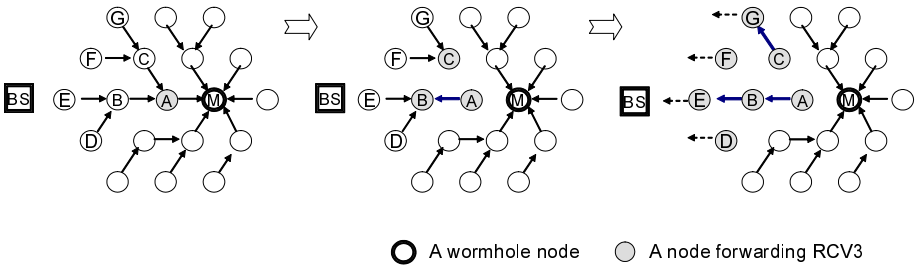


Fig. 1.9. An example of the recovery phase with RCV3 packets

1.5. Simulation Results

This section shows a series of simulation results to evaluate SWAT. SWAT is implemented on TinyOS and simulated in the TOSSIM simulator.[23] In this simulation work, 100 sensor nodes are located in a 10x10 grid topology where the number of physical direct neighbors of each node is eight (Figure 1.10). Node IDs are numbered from 0 to 99 from the left-top corner to the right-bottom corner. The base station is located at the left-top corner (the node 0), and two wormhole links are simulated between wormhole nodes (the nodes 22 and 77, and the nodes 22 and 90) are considered. At the beginning of a simulation, a routing tree is constructed, and the nodes 22 and 77 form a wormhole sphere as shown in Figure 1.1(b). The node 77 attracts data packets from its neighboring nodes and forwards them to the node 22, which in turn forwards them to the node 90. Each node periodically transmits data packets (sensor reading) every 100 seconds. Thus, in average, the base station receives one data packet per second.

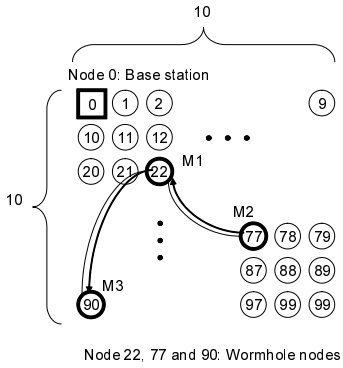


Fig. 1.10. Simulated network

1.5.1. *Performance Implications of SWAT*

Figure 1.11 shows the number of data packets that the base station receives every 15 seconds. (The base station is supposed to receive 15 data packets every 15 seconds.) Due to wormhole nodes, some packets are transmitted to the base station via wormhole links, and others are not. As Figure 1.11 shows, more than half (9 to 11) packets are transmitted via wormhole links. This means that more than half nodes in the network are affected by wormhole nodes. When SWAT starts its self-healing process at the 60th second, it detects and isolates wormhole nodes immediately; the number of data packets transmitted via wormhole links drops to zero. Accordingly, the route recovery phase is successfully executed by the 120th second; all of 15 data packets are transmitted to the base station by avoiding wormhole nodes. Figure 1.11 also shows control overhead in the detection and recovery

phases (gray bars). It is measured by the number of node-to-node transmissions of control packets (danger signals and recovery packets). The total number of control packet transmissions is 101 from the 60th to 120th second. The authors of the paper believe that quick wormhole detection and routing structure recovery compensate for this control overhead.

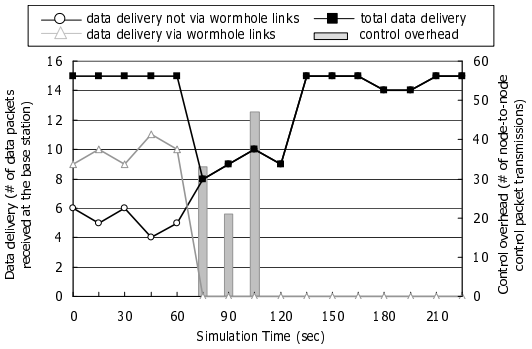
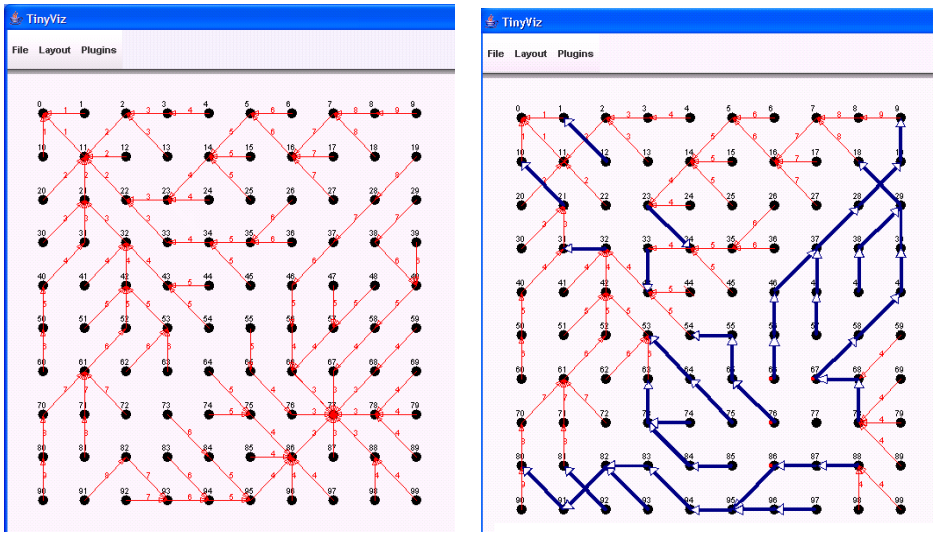


Fig. 1.11. Data delivery and control overhead

Figure 1.12 shows how SWAT self-heals a routing structure distorted by wormhole nodes. Figure 1.12(a) shows a routing structure under a wormhole attack at the 15th second. The nodes 22, 77 and 90 are wormhole nodes as shown in Figure 1.10. Each arrow indicates a routing path across nodes. The node 77 forms a wormhole sphere, and the network is topologically separated. In contrast, Figure 1.12(b) shows a routing structure recovered by SWAT (at the 150th second). Each thick arrow denotes a routing path that SWAT changes in its recovery phase. All three wormhole nodes are successfully isolated; no nodes possess links to wormhole nodes.

Figure 1.13 shows power consumption of control packets in the detection and recovery process. It is measured with power current in mA because the same voltage is used in all nodes. The simulated power current used in the sleep, listening and broadcast states are 5 mA, 10 mA and 25 mA, respectively, as reported by Shnayder et al.[24] The X-Y coordinate represents the network; each cell denotes a node.Figure 1.13(a) shows that power consumption is higher around wormhole nodes (the nodes 22, 77, and 90) due to producing DS1 signals and broadcasting RCV1 packets. Figure 1.13(b) shows that the nodes around the wormhole node 77 consume a higher amount of power than others by producing DS2 signals and broadcasting RCV2 packets. As described in Section 1.4.1, DS2 is specifically designed to detect the wormhole nodes that form wormhole spheres. Figure 1.13(c) shows the power consumption due to DS3 signals and RCV3 packets. Figures 1.13(d) and 1.13(e) show the amount of power spent for DS3 signals and RCV3 packets, respectively. These figures illustrate that DS3 signals are propagated from the edge of a wormhole sphere and DCV3 packets are transmitted throughout the wormhole sphere. As Figure 1.13 demonstrates, the control packets of SWAT works well as they are



(a) Distorted routing structure (b) Recovered routing structure

Fig. 1.12. Recovery of a distorted routing structure

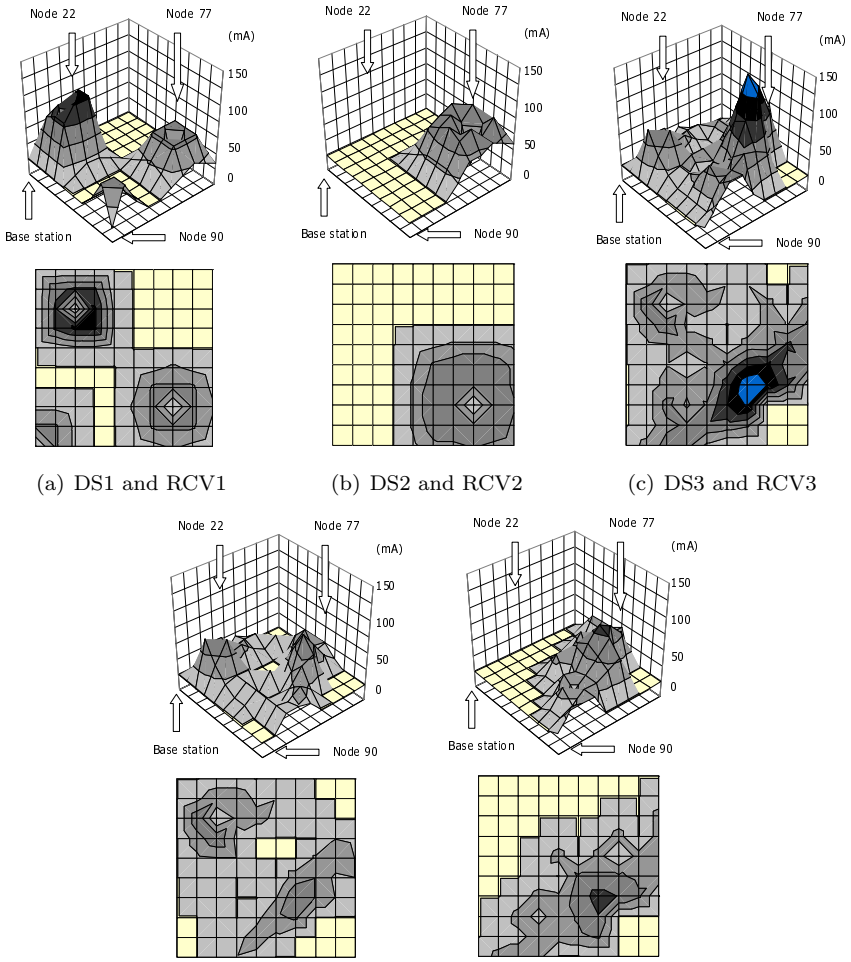
designed.

Figure 1.14 shows how many nodes die over time by running out of power with and without SWAT. As discussed in Sections 1.1 and 1.2, wormhole attacks can shorten the network's lifetime due to packet diversion. Under a wormhole attack, the first dead node is observed at the 440th second, and 51 of 100 nodes are dead at the 600th second. In contrast, using SWAT, the first dead node is observed at the 480th second, and 40 nodes are dead at the 600th second. SWAT successfully expands the network's lifetime by eliminating unnecessary power consumption imposed by wormhole nodes. Figure 1.14 also demonstrates that this benefit compensates well for the power consumption of SWAT shown in Figure 1.13.

Table 1.2 illustrates false isolation in different degrees of network density. False isolation is measured as the number of normal, uncompromised nodes that SWAT incorrectly identify as wormhole nodes and isolate. Network density is represented as the number of physical direct neighbor nodes of each node. As shown in Table 1.2, false isolation decreases as network density grows. In the network density of 10 and 12, no false isolation occurs. However, control overhead (the number of node-to-node control packet transmissions) increases as network density grows.

Table 1.2. False isolation in different degrees of network density

Network density	4	6	8	10	12
False isolation	5	2	1	0	0
Total control overhead	76	89	101	112	115



(a) DS1 and RCV1 (b) DS2 and RCV2 (c) DS3 and RCV3

(d) For DS3 (e) For RCV3

Fig. 1.13. Power consumption of control packets in the detection and recovery phases

Table 1.3 shows the memory footprint of SWAT in a MICA2 mote, and compares it with the footprint of Blink (an example program in TinyOS), which periodically turns on and off an LED. The total memory footprint of SWAT is 1.6 KB in RAM and 11.9 KB in ROM. Of the total ROM footprint, the wormhole detection module occupies 2.0 KB and the recovery module occupies 2.5 KB. The housekeeping module includes neighbor list maintenance and routing tree construction as well as the standard control component in TinyOS, which is responsible of packet transmission and timer control. SWAT is implemented lightweight; it consumes only a small portion of the capacity of a MICA2 mote. (A MICA2 mote has 4 KB in RAM 128 KB in ROM.) SWAT can also run on a smaller-scale mote, for example, TelosB,

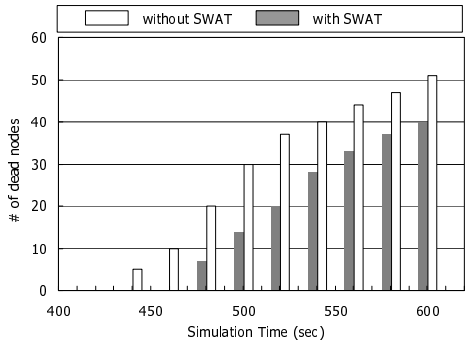


Fig. 1.14. Network lifetime with and without SWAT

Table 1.3. Memory footprint in a MICA2 mote

		RAM (KB)	ROM (KB)
SWAT	Detection module		2.0
	Recovery module		2.5
	House keeping module		7.4
	Total	1.6	11.9
Blink		0.04	1.6

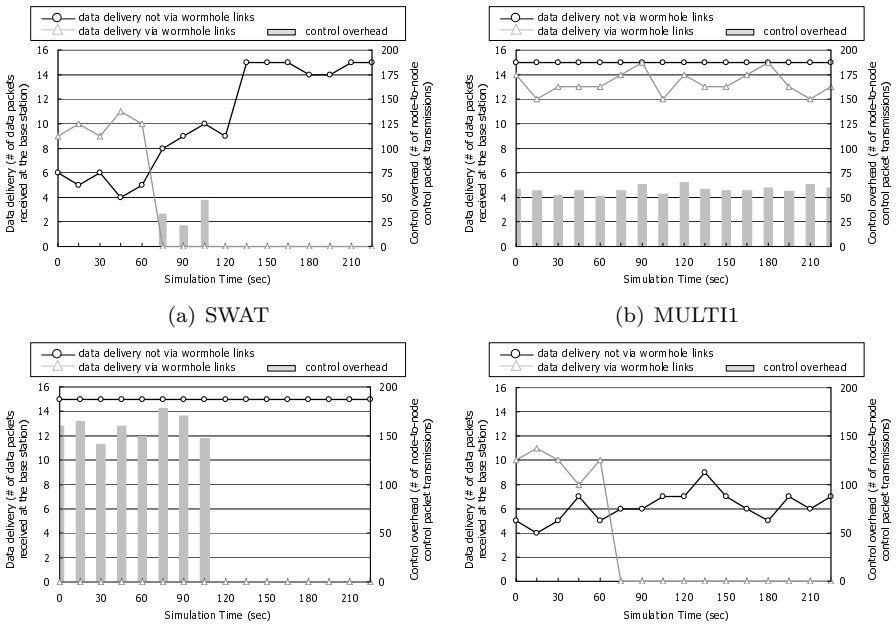
which has 48 KB ROM.

1.5.2. *Performance comparison with other mechanisms*

This section compares SWAT with other wormhole detection mechanisms: flooding-based multipath routing mechanisms (MULTI1[25] and MULTI2[26]) and a connectivity information based mechanism (CONNECT[22]). In MULTI1, each node replicates every incoming data packet and transmits the replicated packets through multiple routing paths toward the base station. In MULTI2, each node reactively discovers multiple paths to the base station per incoming data packet, selects one of them that does not go through wormhole links, and transmits a packet to the selected path. Routing path selection is performed with a statistical analysis method. CONNECT is discussed in Section 1.2.

Figure 1.15 compares data delivery and control overhead in SWAT, MULTI1, MULTI2 and CONNECT. Figure 1.15(a) is same as Figure 1.11. Compared with MULTI1 and MULTI2, SWAT is much less expensive in terms of control overhead. SWAT does not always generate control packets (i.e., danger signals and recovery packets), but generates them only during the detection and recovery phases. Sacrificing control overhead, MULTI1 and MULTI2 guarantee that the base station always receives 15 data packets without using wormhole links. SWAT does not provide this guarantee; however, it performs routing structure recovery so that the base station can receive 15 data packets without using wormhole links.

Both SWAT and CONNECT perform wormhole attack detection and wormhole



(a) SWAT (b) MULTI1

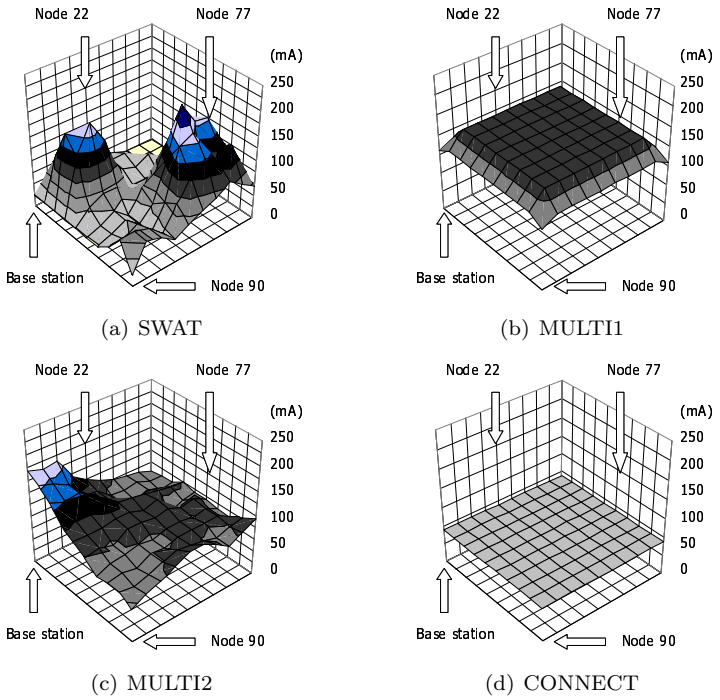
(c) MULTI2 (d) CONNECT

Fig. 1.15. Data delivery and control overhead in SWAT and other related mechanisms

node isolation quickly; the number of data packets transmitted via wormhole links drops rapidly from the 60th to the 70th second. After wormhole node isolation, SWAT recovers distorted routing structure in order to increase the number of data packets that the base station receives. It does not increase in CONNECT because CONNECT does not consider routing structure recovery as discussed in Section 1.2. In terms of control overhead, CONNECT outperforms SWAT. It does not require to transmit control packets across nodes.

Figure 1.16 shows the total power consumption from the 60th to 120th second in SWAT, MULTI1, MULTI2 and CONNECT. Figure 1.16(a) indicates that, in SWAT, the nodes around wormhole nodes spent more powers than the other nodes in order to generate/transmit danger signals and recovery packets. SWAT does not impose unaffected nodes to consume power. Figure 1.16(b) shows all nodes consume a very similar amount of power in MULTI1. Figure 1.16(c) shows the nodes around the base station consume more power than the other nodes. Power consumption in MULTI1 and MULTI2 are quite high due to their high control overhead shown in Figure 1.15. Both mechanisms incur unnecessary power consumption to unaffected nodes. Power consumption in CONNECT is very low because it does not require to transmit control packets across nodes (Figure 1.15).

Table 1.4 shows the false detection and isolation in SWAT and CONNECT. False detection is measured as the number of normal, uncompromised nodes that



(a) SWAT (b) MULTI1

(c) MULTI2 (d) CONNECT

Fig. 1.16. Total power consumptions in SWAT and other related mechanisms

are incorrectly identified as wormhole nodes. False isolation is measured as the number of normal, uncompromised nodes that are incorrectly isolated from the network. As discussed in Section 1.5.1, false detection and isolation decrease in SWAT as network density (the number of physical direct neighbors of each node) grows. In CONNECT, false isolation increases as network density grows, although false detection remains zero. In the network density of 12, CONNECT incorrectly isolates more than 30% nodes (36 nodes) in the network. As described in Section 1.2, CONNECT cannot pinpoint the locations of wormhole nodes and isolates several normal, uncompromised nodes around wormhole nodes.

Table 1.4. False detection and false isolation in different degrees of network density

		Network density				
		4	6	8	10	12
SWAT	False detection	5	2	1	0	0
	False isolation	5	2	1	0	0
CONNECT	False detection	0	0	0	0	0
	False isolation	12	18	24	30	36

1.6. Conclusion

This paper proposes and evaluates a decentralized self-healing mechanism, SWAT, which detects and recovers from wormhole attacks in wireless sensor networks. Upon detecting a wormhole attack, SWAT identifies the locations of wormhole nodes, isolates them from the network and recovers the routing structure distorted by them. SWAT is the first mechanism that performs both wormhole node isolation and routing structure recovery against wormhole attacks. Unlike many other wormhole detection mechanisms, SWAT does not require any extra networking facilities (e.g., timing analysis and localization facilities) as well as special hardware (e.g., GPS). Instead, it uses network connectivity information only in a decentralized manner. Simulation results show that SWAT yields 100% wormhole attack detection, 0% false detection, 100% wormhole node isolation and 0% false isolation in dense networks. The results also show that SWAT outperforms multi-path routing mechanisms in terms of control overhead and power consumption and outperforms another connectivity-based detection mechanism in terms of false isolation rate and recovery efficiency. SWAT is implemented lightweight so that it can operate on resource-limited sensor nodes such as MICA2 and TelosB nodes.

Future research directions are to make SWAT more robust against a wider range of network topology, network density and node mobility configurations. Moreover, SWAT will be empirically evaluated with real sensor nodes.

References

1. A. Perrig, J. Stankovic, and D. Wagner, Security in wireless sensor networks, *Communications of the ACM.* **47**(6) (June, 2004).
2. H. Alzaid, S. Abanmi, S. Kanhere, and C. Chou. Detecting wormhole attacks in wireless sensor networks. Technical report, Department of Computer Science and Engineering, University of New South Wales (October, 2006).
3. M. A. Gorlatova. Review of existing wormhole attack discovery techniques. Technical report, School of Information Technology and Engineering, University of Ottawa (Augast, 2006).
4. Y. Hu, A. Perrig, and D. Johnson. Wormhole detection in wireless ad hoc network. Technical report, Department of Computer Science, Rice University (June, 2002).
5. C. Karlof and D. Wagner. Secure routing in sensor networks: Attacks and countermeasures. In *Proc. of First IEEE International Workshop on Sensor Network Protocols and Applications* (May, 2003).
6. Y. Hu, A. Perrig, and D. Johnson. Packet leashes: A defence against wormhole attacks in wireless networks. In *Proc. of the Twenty-second IEEE International Conference on Computer Communications* (April, 2003).
7. S. Capkun, L. Buttyan, and J. Hubaux. Sector: Secure tracking of node encounters in multi-hop wireless networks. In *Proc. of the First ACM Workshop on Security of Ad Hoc and Sensor Networks* (October, 2003).
8. X. Wang and J. Wong. An end-to-end detection of wormhole attack in wireless ad-hoc networks. In *Proc. of the Thirty-first Annual International Computer Software and Applications Conference* (July, 2007).

9. L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. In *In Network and Distributed System Security Symposium* (Feburary, 2004).
10. N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *Proc. of the ACM Workshop on Wireless Security* (September, 2003).
11. D. L. Mills. A computer-controlled loran-c receiver for precision timekeeping. Technical report, Department of Electrical and Computer Engineering, University of Delaware (March, 1992).
12. D. L. Mills. A precision radio clock for wwv transmissions. Technical report, Department of Electrical and Computer Engineering, University of Delaware (August, 1997).
13. J. Zhen and S. Srinivas. Preventing replay attacks for secure routing in ad hoc. networks. In *Proc. of the Second International Conference on ADHOC Networks and Wireless* (October, 2003).
14. R. Poovendran and L. Lazos, A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks, *ACM Journal of Wireless Networks.* **13**(1), 27–59 (Januaray, 2005).
15. L. Lazos and R. Poovendran. Serloc: Secure range-independent localization for wireless sensor networks workshop on wireless security archive. In *Proc. of the Third ACM Workshop on Wireless Security* (October, 2004).
16. I. Khalil, S. Bagchi, and N. B. Shroff. Liteworp: A lightweight countermeasure for the wormhole attack in multihop wireless network. In *Proc. of International Conference on Dependable Systems and Networks* (October, 2005).
17. T. Phuong, L. Hung, S. Cho, Y. Lee, and S. Lee. An anomaly detection algorithm for detecting attacks in wireless sensor networks. In *Proc. of the Fourth IEEE Intelligence and Security Informatics Conference* (May, 2006).
18. I. Vajda, L. Buttyan, and L. Dora. Statistical wormhole detection in sensor networks. In *Proc. of the Second European Workshop: Security and Privacy in Ad-hoc and Sensor Networks* (July, 2005).
19. Y. Hou, C. Chen, and B. Jeng. Distributed detection of wormholes and critical links in wireless sensor networks. In *Proc. of the Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing* (November, 2007).
20. W. Wang and A. Lu, Interactive wormhole detection and evaluation, *Information Visualization.* **6**(1), 3–17 (January, 2007).
21. Y. Xu, G. Chen, J. Ford, and F. Makedon. Distributed wormhole attack detection in wireless sensor networks. In *Proc. of the First Annual IFIP Working Group 11.10 International Conference on Critical Infrastructure Protection* (March, 2007).
22. R. Maheshwari, J. Gao, and S. R. Das. Detecting wormhole attacks in wireless networks using connectivity information. In *Proc. of the Twenty-sixth IEEE International Conference on Computer Communications* (May, 2007).
23. P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proc. of the First ACM Conference on Embedded Networked Sensor Systems* (November, 2003).
24. V. Shnayder, M. Hempstead, B. Chen, G. Werner-Allen, and M. Welsh. Simulating the power consumption of large scale sensor network applications. In *Proc. of the Second ACM Conference on Embedded Networked Sensor Systems* (November, 2004).
25. S. Berton, H. Yin, C. Lin, and M. G. Secure, disjoint, multipath source routing protocol for mobile ad-hoc networks. In *Proc. of the Fifth International Conference on Grid and Cooperative Computing* (October, 2006).
26. L. Qian, N. Song, and X. Li, Detection of wormhole attacks in multi-path routed wireless ad hoc networks: a statistical analysis approach, *Journal of Network and Computer Application.* **30**(1), 308–330 (January, 2007).