When Ants Attack: Security Issues for Stigmergic Systems

Weilin Zhong and David Evans
University of Virginia, Department of Computer Science
[weilin, evans]@virginia.edu

Abstract

Stigmergic systems solve global problems by using indirect communication mediated by an environment. Because they are localized and dynamic, stigmergic systems are self-organizing, robust and adaptive. These properties are useful for creating survivable systems, but stigmergic systems also raise new security concerns. Indirect communication makes systems more vulnerable in an open and hostile environment, and feedback mechanisms common to stigmergic algorithms can be exploited by attackers. In this paper we use AntNet, an adaptive routing algorithm inspired by biological ant foraging, to explore some of the security issues for stigmergic systems. We identify possible attacks and analyze their potency. We propose and evaluate mechanisms for defending against these attacks.

1 Introduction

Large-scale complex distributed systems, such as peer-to-peer Intenet applications, wireless ad hoc networks and distributed sensor networks present both new problems and opportunities. *Stigmergic systems* build these applications by using indirect communication mediated by a shared environment [Grassé59]. These systems involve many independent units interacting to achieve global tasks.

Biology provides much inspiration for the design of robust systems [Bonabeau99]. Social insects are well known for their complex group behaviors emerging from the cooperative behaviors of many small and simple members. Without any leader or centralized control, an insect colony solves problems that are far beyond the capability of any individual insect, such as finding food or building nest. The intelligence of social insect groups lies in their interactions, which are achieved by altering and sensing the concentration of chemical pheromones in the common environment. Biologically inspired algorithms have been developed for many problems including communication network routing problems [DiCaro98a, DiCaro98b, Scho96, White97], distributed intrusion detection and response [Fenet01], graph exploration [Yanovski01], terrain coverage [Koenig01] and designing peer-to-peer applications [Montresor01]. All of these use independent agents interacting in a common environment to achieve global properties.

The adaptability and decentralization of stigmergic systems make them intrinsically resilient to certain classes of attacks. Several researchers have argued for the survivability of distributed, decentralized systems. Fisher and Lipson propose using *emergent algorithms* as a way to build survivable systems and present an Internet routing protocol based on this approach [Fisher98]. They define emergent algorithms as any computation that achieves predictable global effects by communication directly with only a bounded number of immediate neighbors and without any central control or global knowledge. Stigmergic algorithms are a class of emergent algorithms.

On the other hand, stigmergic systems offer malicious intruders opportunities to wreak havoc not available with traditional systems. To our knowledge, this is the first work to study security vulnerabilities of stigmergic systems. In this paper, we study the security issues and opportunities for AntNet [DiCaro98a, DiCaro98b], an adaptive routing algorithm inspired by the stigmergic model of ant foraging behaviors. The next section describes AntNet. Section 3 analyzes vulnerabilities of AntNet and identifies three classes of attacks. The rest of the paper analyzes those attacks and suggests defensive mechanisms.

2 AntNet

AntNet [DiCaro98a, DiCaro98b] is an adaptive routing algorithm inspired by ant colonies. An AntNet node maintains probabilistic entries in the routing table, indicating the *goodness* of each output link for each destination. AntNet nodes periodically send out mobile agents known as *ant packets* to explore paths to a specified destination. There are two kinds of ant packets: *forward ants* and *backward ants*. Forward ants explore the network to find a feasible and low-cost path, recording every node it visits. Once it arrives at the destination, it is converted into a backward ant. The backward ant returns to the source node following the path in reverse. At each node, it measures the goodness of this path based on ant trip time and increases the corresponding goodness values associated with the link that it comes from in the local routing table.

Ants interact and communicate indirectly by updating the routing tables, thus collaboratively solve the global network routing optimization problem. With ants continuously collecting path information and exploring new paths, AntNet is able to adapt to changes in network topology and traffic load. The decentralized and adaptive nature of AntNet make it tolerate to faults and resilient under traditional network attacks.

3 Vulnerabilities and Attacks

Much work has been done on the vulnerabilities, attacks and defenses on traditional routing protocols. [Vetter97, Perlman88, Zhang98, Hauser99]. We focus on the new vulnerabilities particular to the stigmergic properties of AntNet.

AntNet includes no mechanisms to protect and verify the routing information carried by the ant agents. Nodes completely trust information in all backward ants they receive and update their routing tables accordingly. AntNet would be vulnerable to various attacks when operating in a hostile environment. In particular, we consider threats due to a compromised node. Similar threats would result if a link was compromised and an intruder was able to inject or tamper with ant packets on the wire. We assume that a node subverted by an intruder can monitor, fabricate, replay, modify and delete ant packets. The routing information itself is not considered confidential so we ignore routing information disclosure threats and focus on integrity.

A successful attack perturbs the goodness values in routing tables of other nodes. These attacks lead to changes in packet latency and throughput. The three most basic attack mechanisms are to: fabricate ant packets, drop ant packets, and tamper with information in ant packets.

3.1 Fabrication Attacks

An attacker who compromises a node or link can inject fabricated ant packets into the system or replay observed ants. We simulate this attack using the network topology shown in Figure 1. We use the simple network topology shown in Figure 1. Each link is bi-directional and all link parameters (transmission rate and propagation delay) are identical. The network flows and ants are generated by node 0 only and destined for node 4. Ants are generated every 100ms and contain 192 bits. Data packets are generated at a constant rate with an average size of 1024 bits. We simulate AntNet using OMNET++ [Varga01].

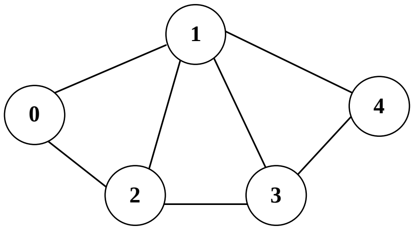

Figure 1. Experimental Network

A subverted node 2 begins generating bogus ants at the 200th second. It injects ten bogus backward ants for every incoming backward ant to falsely promote the link 0→2. The bogus ant has a path 0234 and a trip time 4.8ms, which is the optimal trip time of path 0234 without congestion. We can see from Figure 2 that node 2 can easily deceive node 0 into believing link 0→2 was the best link towards node 4.

The easiest way to defend against a fabricated or replayed ant attack is to simply record legitimate ant packets by assigning them unique identifiers. We can uniquely identify each ant by the tuple <*source*, *id* > where source uniquely identifies the node that generates the ant and the unique *id* is a local counter on the generating node. Each node maintains a list of all passing forward ants and only accepts those backward ants whose identity tuple is contained in that list. Once a legitimate backward ant arrives, its identifier is deleted from the list, thereby preventing replay attacks. Backward ants that do not have a valid ID are dropped and ignored. Entries in the list expire and are removed if a corresponding backward ant does not arrive within a threshold time. Figure 3 demonstrates that the ant ID mechanism effectively defends AntNet from a bogus attack.

3.2 Dropping Attacks

Dropping ant packets is not easy to detect and is often indistinguishable from real network failure. But the effectiveness of dropping ant attacks is limited by the location of the compromised node.

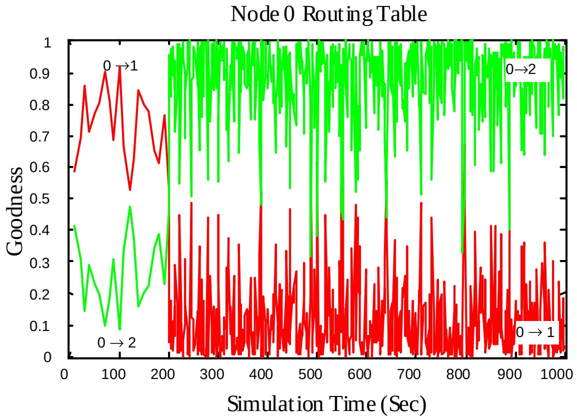

Figure 2. Node 2 generates bogus ants with path 0234 starting from 200th sec. Link 0→2 achieves goodness value higher than threshold value 0.7 soon and becomes the chosen path.

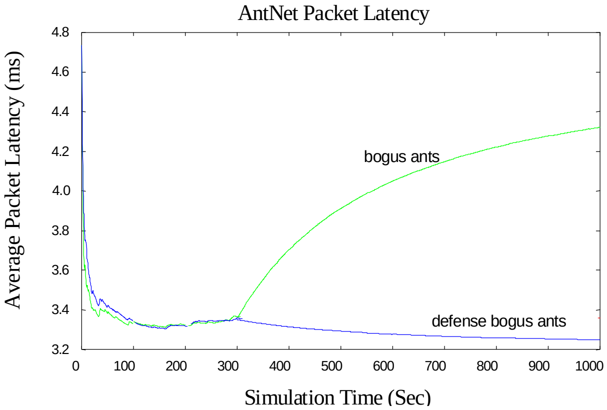

Figure 3. Average packet latency of two scenarios: bogus attack, and attack with defense method.

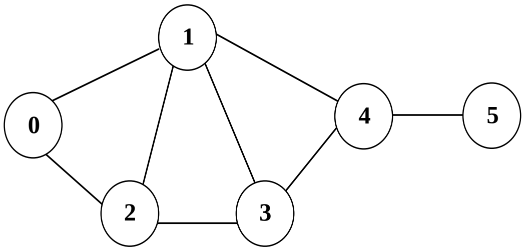

Figure 4. Network topology for drop ant packets experiment.

When an attacker subverts a node on the best path between two points, it can discredit good paths by selectively dropping ant packets. If the compromised node is not on the best path, dropping ant packets is not an effective attack, since it would just repel the traffic from this path, which would make AntNet find the best path more quickly.

To illustrate how dropping ant packets can be effective, we use a revised network topology shown in Figure 4 with both data and ants flowing from node 0 to node 5. We simulate the attack where node 4 selectively drops ant packets that have visited node 1. Under this attack, path 014 is not reinforced and will soon be abandoned. Data packets will all be routed through path 0234. The average packet latency under attack compared with packet lantency without attack is shown in Figure 5. Node 4 could use this attack to usurp the link between itself and node 1. Instead of just harming another node, this attack directly benefits the attacker by removing competition for network resources.

4 Tampering Attacks

A backward ant records the path trip time by maintaining the sum of the local link latency estimated along the reverse path. Beginning at the destination node, the trip time is set to 0. When a backward ant arrives at a node *x* coming from an adjacent node *y*, the link latency $L(x, y)$ of the link $x \longrightarrow y$ is estimated based on the local workload and queue length and added to the trip time $T_{y \to dest}$ carried by this ant to get $T_{x \to dest} = L(x, y) + T_{y \to dest}$. When a backwards ant reaches the source node, the whole trip time of source node to destination node, $T_{src \to dest}$ is known.

Given network topology as in Figure 1 using setup described in 3.1, the uncongested single link latency is 1.6ms. Suppose a malicious attacker compromises node 2 and can tamper with passing backward ants, setting up the trip time $T_{2 \to 4}$ to 0ms (or a negligibly small value; a negative value may also be possible but can be easily detected). At node 0 the trip time of path 0234 is calculated as 1.6ms instead of 4.8ms, making path 0234 appear faster than path 014. The experiment shown in Figure 6

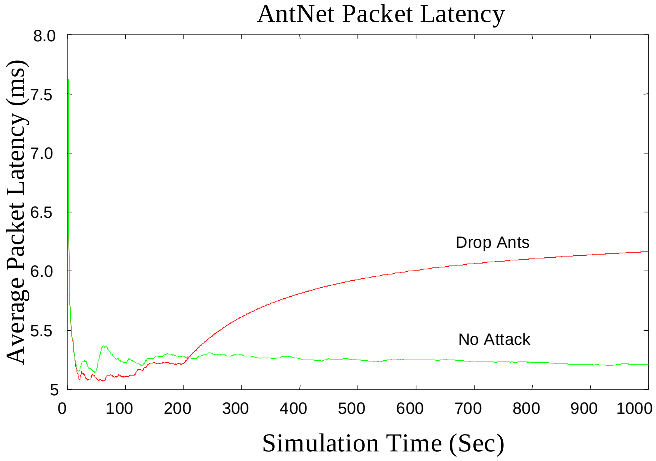

Figure 5. Average Packet Latency. Node 4 drops forward ants that have visited node 1. Packet latency under attack will soon approach 6.4ms, the trip time of path 02345.

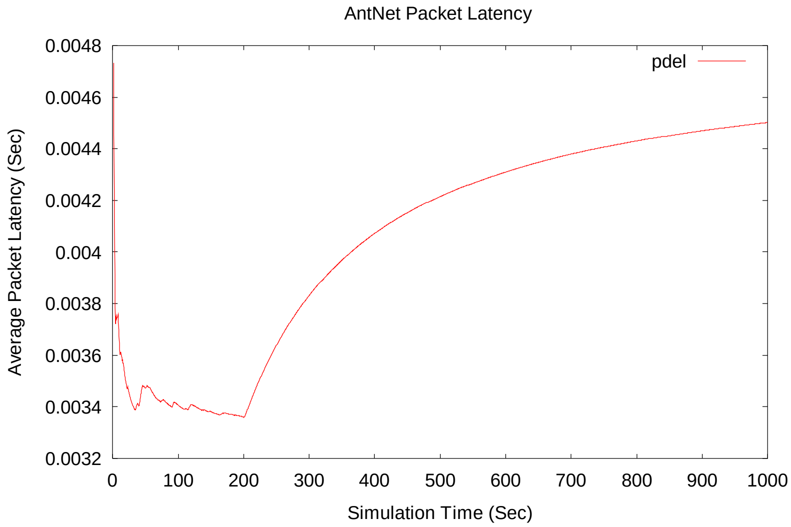

Figure 6. Tampering Attack Effectivness. Node 2 tampers with passing ants stating as time 200s. Packet latency gradually increases from 3.2ms (path 014) to 4.8ms (path 0234).

illustrates what happens when node 2 tampers with the trip time information carried by passing ants from 200th second in the simulation. This attack is effective in switching the routing probabilities at node 0 to direct all data packets through the compromised node. Next we discuss the locality properties of AntNet which help to limit the effectiveness of tampering attacks, and then consider some cryptographic defenses.

4.1 Locality of Damages

Consider an attacker who wants to prevent data packets from being routed through the best path. A compromised node on the best path between source and destination nodes can easily achieve that by lying that the best path takes extremely long time. Hence, we only consider compromised node not along the best path. In this case, the attacker's goal is to attract packets towards that inferior path. The attacker can achieve this in two ways: modifying trip time information carried by ants or injecting bogus ants to promote this path. As discussed in Section 3.1, bogus ants can be easily detected using ant identifiers, so the only attack likely to succeed is to modify recorded trip times. We should see that the capability of a malicious node to succeed with the tampering attack is highly location dependent.

When a backward ant arrives at each node, the path latency from current node to the ant's destination is estimated by adding locally estimated link latency to the trip time information carried by this backwards ant. A malicious node can alter the trip time from itself to the target destination. The maximal lie is to record the trip time as 0. The malicious node can do nothing to alter the trip time for the remaining part of the path.

Consider the network in Figure 7. Suppose link latency is 1 for all links. The best path is 014 with a trip time 2. The maximal lie that malicious node 2 can tell is that the trip time from itself to node 4 is 0. So malicious node 2 can make node 0 believe that path 0234 has a trip time 1, which is better than path 014. Instead, if node 3 were malicious, it could not tamper with the trip time on the

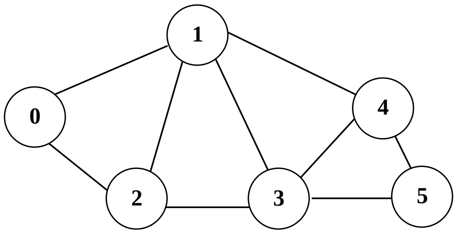

Figure 7. Network topology for locality.

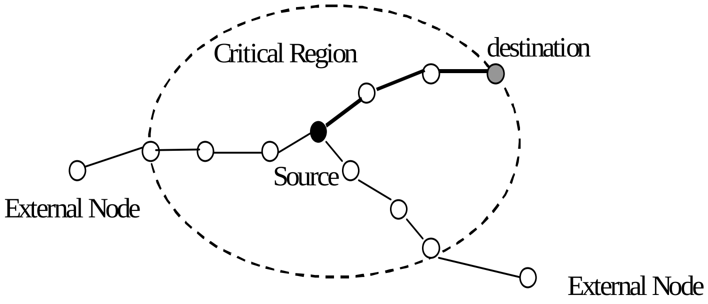

Figure 8. Critical region. The shortest path is 3 hops. With no congestion, nodes further than 3 hops away from source are external to the critical region between source and destination.

return path from 3→0, which has a trip time 2, so it can only make path 0234 as good as path 014, hence, 50% of network traffic will go through this link. A compromised node 5 can do nothing to make the path 02354 better than the best path 014.

Intuitively, those nodes that have a distance coming from source node *S* shorter than the shortest distance from *S* to *D* can perturb flows between a source and destination node by tampering attacks. They form a *critical region*:

> In a network *N*, the critical region for *S* and *D* is the set of all nodes *x* where there is a path *S*→*x* with latency less than or equal to the latency of the best path *S*→*D*.

We can visualize the critical region as a circle centered at source node *S*, with a radius of the best path latency shown in Figure 8. Since the latency of the best path changes according to network situation, membership in the critical region is dependent on current network conditions.

In order to carry out effective tampering attacks on a pair of notes, a compromised node must be in the critical region. An external node cannot influence the traffic flows between the source and destination node by tampering, although it could attempt to increase the latency (and hence, the size of the critical region) by network flooding attacks. An external node may be able to introduce some traffic congestion on the best path to enlarge the radius of the critical region to include itself. This attack requires the external node to transmit a large number of packets, and is highly dependent on the network topology and location of malicious node. Once the critical region has grown to include the external node, it must continue to create the network congestion otherwise the routing information will quickly recover to the original best path.

4.2 Mitigating Tampering Attacks

Because of the locality properties, we need only defend against tampering attacks from nodes within the critical region. One way to defend against tampering attacks is to use cryptographic to verify the integrity of path information carried by ants.

Public key signatures have been used to provide authenticity and integrity for traditional routing protocols [Murphy96, Smith96]. Link state routing algorithms, such as OSPF, flood routing information to the whole network, so the source of the routing information can be determined. Thus public key signatures can provide the required source authentication and integrity. However, in distance-vector routing algorithms, nodes integrate and propagate routing updates so the original source of information cannot be determined.

AntNet is similar to distance vector routing protocols in that a node possesses only knowledge of local links and receives summaries of path latency information from its neighbors. One proposed solution secures distance-vector routing protocol by adding predecessor information in routing updates and using path traversal mechanisms to verify the whole path [Smith96].

We can use similar technique to protect AntNet from tampering attacks as shown in Figure 9. Assume each node has a public-private key pair whose public key can be securely obtained by all other nodes (note that the critical region limits the number of nodes to consider). Along the forward path, every node signs the successor it chooses to prevent tampering of the path information. The signed proof contains the ant identifier *n*, the current node's identity, and the identity of its successor. The destination node verifies the integrity of the whole path and signs the complete path. Assuming the destination node is trustworthy, the signed proofs generated by intermediate nodes can be removed from backward ant. On the returning path, each node signs the locally estimated link latency. The source node verifies the path by verifying the signature of destination node and verifies the trip time by verifying link latencies signed by all the intermediate nodes. The tampering of trip time is limited: a malicious node can only lie about the delay of the link between itself and its successor.

The cryptographic technique discussed above is too expensive for many applications in which we wish to use stigmergic algorithms. It requires public key operations at every node on the path for both directions. A less expensive but less robust technique is to use *verification ants* whenever the goodness of a link goes up to the threshold. Before change the routings, the node sends out a verification ant containing a nonce to measure the trip time required to reach the destination node. The verification ant should go through the data queues and follows the best links along all intermediate nodes, trying to reach destination node as soon as possible. The destination node signs the verification ant and sends it back to the source node. The source node only cares whether a link should be set up as the best link for specific destination, so it only wants to test whether going from the tested link can reach the destination within the ant estimated trip time. Therefore, source node does not care which path this test ant actually goes through as long as it reaches the destination node coming and returning from the tested link. There are two options for calculating the one-way trip time from the source to the destination. The first option is to assume the time is similar in both directions, and just estimate the one-way trip time is half of the round-trip time. This assumption is often unrealistic for typical networks. The second option is to assume clock synchronization among nodes. Clock synchronization can be achieved by some time synchronization protocols or GPS service [Tanenbaum95]. In this case, the destination node includes a signed timestamp in the return packet.

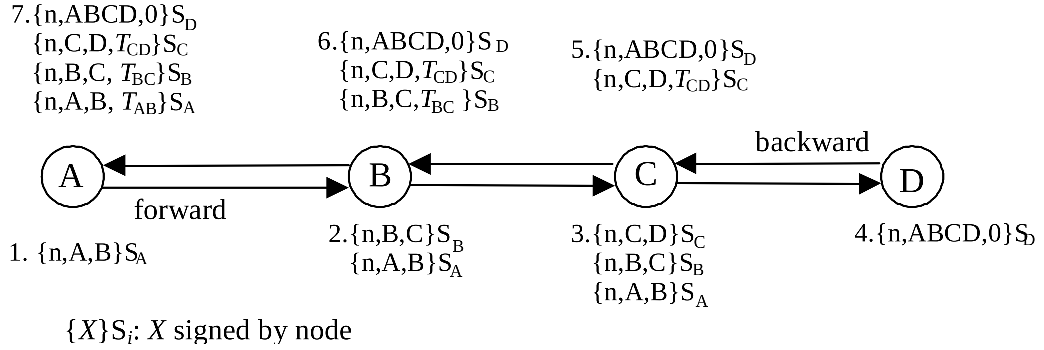

Figure 9. Digital Signature Path Authentication

5 Summary

Stigmergic systems offer new opportunities for building secure systems by taking advantage of decentralized control and indirect communication. The decentralized and adaptive properties of AntNet, typical of stigmergic systems, provide desirable security properties without any cryptographic mechanisms. AntNet can be made resilient to all except on of the attack classes we identified with simple, lightweight mechanisms. The exception was tampering attacks, but locality properties limit the need for cryptographic defenses.

Our work is a first step towards understanding some of the security risks associated with stigmergic systems. As these types of systems become more widely used to develop survivable and adaptive systems, it is important that the new vulnerabilities they introduce are also considered.

Acknowledgements

This work was funded in part by the National Science Foundation (CCR-0092945). Gianni Di Caro provided source code to an implementation of AntNet, and graciously assisted us in adapting it for our experiments.

References

[Bonabeau99] Eric Bonabeau, Marco Dorigo, Guy Theraulaz. *Swarm Intelligence: from Natural to Artificial Systems*, Santa Fe Institute, Oxford University Press, 1999.

[DiCaro98a] Gianni Di Caro, *AntNet: Distributed Stigmergic Control for Communications Networks*, Journal of Artificial Intelligence Research 9 (1998): 317-365.

[DiCaro98b] Gianni Di Caro. *Two Ant Colony Algorithms for Best-effort Routing in Datagram Networks.* 10th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'98). IASTED/ACTA Press, 1998.

[Fenet01] Serge Fenet, Salima Hassas. *A distributed Intrusion Detection and Response System Based on mobile autonomous agents using social insects communication paradigm*. First International Workshop on Security of Mobile Multiagent Systems, 2001.

[Fisher98] David A. Fisher and Howard F. Lipson. *Emergent Algorithms - A New Method for Enhancing Survivability in Unbounded Systems*. Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences. 1998.

[Grassé59] Grassé, P.-P. *La Reconstruction du nid et les Coordinations Inter-Individuelles chez Bellicositermes Natalensis et Cubitermes sp. La théorie de la Stigmergie:Essai d'interprétation du Comportement des Termites Constructeurs.* Insect. Soc.6 (1959).

[Hauser99] Ralf Hauser, Tony Przygienda and Gene Tsudik. *Lowering security overhead in link state routing.* Computer Networks, Volume 31 Number 8. 1999.

[Koenig01] Sven Koenig, B. Szymanski and Y. Liu. *Efficient and Inefficient Ant Coverage Methods*. Annals of Mathematics and Artificial Intelligence. Vol 31, Issue 1/4. 2001.

[Kong01] Jiejun Kong, Petros Zerfos, Haiyun Luo, Songwu Lu, Lixia Zhang. *Providing Robust and Ubiquitous Security Support for Mobile Ad-hoc Networks*. IEEE 9th International Conference on Network Protocols (ICNP). November 2001,

[Kurose01] James F. Kurose, Keith W.Ross. *Computer Networking: A Top-Down Approach Featuring the Internet.* Addison Wesley, 2001

[Montresor01] Alberto Montresor. *Anthill: a Framework for the Design and Analysis of Peer-to-Peer Systems.* 4th European Research Seminar on Advances in Distributed Systems. May 2001.

[Murphy96] S. Murphy and M. Badger. *Digital Signature Protection of the OSPF Routing Protocol*. Internet Society Symposium on Network and Distributed Systems Security, 1996.

[Perlman88] R. Perlman. *Network Layer Protocol with Byzantine Agreement.* MIT PhD Thesis (available as MIT LCS TR-429). October 1998.

[Scho96] R.Schoonderwoerd, O.Holland, J.Bruten and L.Rothkrantz. *Ant-based load balancing in telecommunications networks.* Adapt. Behav. 5 (1996): 169-207.

[Smith96] Bradley R.Smith, Shree Murthy, J.J. Garcia-Luna-Aceves. *Securing Distance-Vector Routing protocols.* IEEE/ISOC Symposiums on Network and Distributed System Security, 1996.

[Tanenbaum95] Andrew S. Tanebaum, "Distributed Operating Systems," pp.118 – 133, 1995, Prentice-Hall, Inc.

[Varga01] András Varga. *OMNeT++: a discrete event simulation tool*. http://www.hit.bme.hu/phd/vargaa/omnetpp.htm.

[Vetter97] Brian Vetter, Feiyi Wang and S. Felix Wu. *An Experimental Study of Insider Attacks for the OSPF Routing Protocol*. 5th IEEE International Conference on Network Protocols, Atlanta, GA. IEEE press, October 1997.

[White97] T. White. *Routing with swarm intelligence*. Technical Report SCE97-15, Systems and Computer Engineering Department, Carleton University, September, 1997.

[Yanovski01] Vladimir Yanovski, Israel A. Wagner, Alfred M. Bruckstein. *Computer Vertex-Ant-Walk – A robust method for efficient exploration of faulty graph.* Annals of Mathematics and Artificial Intelligence. Volume 31, Issue 1/4. 2001.

[Zhang98] Kan Zhang. *Efficient Protocols for Signing Routing Messages*. Symposium on Network and Distributed Systems Security (NDSS). 1998.