FINDING OF PACKET SINKING SPASMS IN WIRELESS AD HOC NETWORKS

[1]P.Veeramuthu, [2] Makkineni manoj
[1]Assistant Professor, [2]Student
[1]Computer science Department,
[1]B.T College, Madanapalle, India.

Abstract : ***The most of the related works preclude the ambiguity of the environment by assuming that malicious sinking is the only source of packet loss, so that there is no need to account for the impact of link errors. On the other hand, for the small number of works that differentiate between link errors and malicious packet drops, their detection algorithms usually require the number of maliciously-dropped packets to be significantly higher than link errors, in order to achieve an acceptable detection accuracy. Our algorithm also provides a truthful and publicly verifiable decision statistics as a proof to support the detection decision. The high detection accuracy is achieved by exploiting the correlations between the positions of lost packets, as calculated from the auto-correlation function of the packet-loss bitmap a bitmap describing the lost or received status of each packet in a sequence of consecutive packet transmissions. The basic idea behind this method is that even though malicious dropping may result in a packet loss rate that is comparable to normal channel losses, the stochastic processes that characterize the two phenomena exhibit different correlation structures. Therefore, by detecting the correlations between lost packets, one can decide whether the packet loss is purely due to regular link errors, or is a combined effect of link error and malicious drop.***

Index Terms - packet dropping, auto correlation function, secure routing, attack detection, homomorphic linear signature, auditing.

I. INTRODUCTION

Link error and malicious packet dropping are two sources for packet losses in multi-hop wireless ad hoc network. While observing a sequence of packet losses in the network, we are interested in determining whether the losses are caused by link errors only, or by the combined effect of link errors and malicious drop. Even though persistent packet dropping can effectively degrade the performance of the network, from the attacker's standpoint such always on attack has its disadvantages. First, the continuous presence of extremely high packet loss rate at the malicious nodes makes this type of attack easy to be detected. Second, once being detected, these attacks are easy to mitigate. For example, in case the attack is detected but the malicious nodes are not identified, one can use the randomized multi-path routing algorithms to circumvent the black holes generated by the attack, probabilistically eliminating the attacker's threat. If the malicious nodes are also identified, their threats can be completely eliminated by simply deleting these nodes from the network's routing table. A malicious node that is part of the route can exploit its knowledge of the network protocol and the communication context to launch an insider attack—an attack that is intermittent, but can achieve the same performance degradation effect as a persistent attack at a much lower risk of being detected. Specifically, the malicious node may evaluate the importance of various packets, and then drop the small amounts that are deemed highly critical to the operation of the network. For example, in a frequency-hopping network, these could be the packets that convey frequency hopping sequences for network-wide frequency-hopping synchronization; in an ad hoc cognitive radio network, they could be the packets that carry the idle channel lists that are used to establish a network-wide control channel. By targeting these highly critical packets, the authors in have shown that an intermittent insider attacker can cause significant damage to the network with low probability of being caught. We are interested in combating such an insider attack. In particular, we are interested in the Detection of packet dropping attacks in wireless ad hoc networks.

Problem of detecting the occurrence of selective packet drops and identifying the malicious node(s) responsible for these drops. Detecting selective packet-dropping attacks is extremely challenging in a highly dynamic wireless environment. The difficulty comes from the requirement that we need to not only detect the place where the packet is dropped, but also identify whether the drop is intentional or unintentional. Specifically, due to the open nature of wireless medium, a packet drop in the network could be caused by harsh channel conditions or by the insider attacker. In an open wireless environment, link errors are quite significant, and may not be significantly smaller than the packet dropping rate of the insider attacker. So, the insider attacker can camouflage under the background of harsh channel conditions. In this case, just by observing the packet loss rate is not enough to accurately identify the exact cause of a packet loss. The above problem has not been well addressed in the literature. Most of the related works preclude the ambiguity of the environment by assuming that malicious dropping is the only source of packet loss, so that there is no need to account for the impact of link errors. On the other hand, for the small number of works that differentiate between link errors and malicious packet drops, their detection algorithms usually require the number of maliciously-dropped packets to be significantly higher than link errors, in order to achieve acceptable detection accuracy. We develop an accurate algorithm for detecting selective packet drops made by insider attackers. Our algorithm also provides a truthful and publicly verifiable decision statistics as a proof to support the detection decision. The high detection accuracy is achieved by exploiting the correlations between the positions of lost packets, as calculated from the auto-correlation function (ACF) of the packet-loss bitmap—a bitmap describing the lost/received status of each packet in a sequence of consecutive packet transmissions. The basic idea behind this method is that even though malicious dropping may result in a packet loss rate that is comparable to normal channel losses, the stochastic processes that characterize the two phenomena exhibit different correlation structures. Therefore, by detecting the correlations between lost packets, one can decide whether the packet loss is purely due to regular link errors, or is a combined effect of link error and malicious drop.

Our algorithm takes into account the cross-statistics between lost packets to make a more informative decision, and thus is in sharp contrast to the conventional methods that rely only on the distribution of the number of lost packets. The main challenge in our mechanism lies in how to guarantee that the packet-loss bitmaps reported by individual nodes along the route are truthful, i.e., reflects the actual status of each packet transmission. Such truthfulness is essential for correct calculation of the correlation between lost packets. This challenge is not

trivial, because it is natural for an attacker to report false information to the detection algorithm to avoid being detected. For example, the malicious node may understate its packet-loss bitmap, i.e., some packets may have been dropped by the node but the node reports that these packets have been forwarded. Therefore, some auditing mechanism is needed to verify the truthfulness of the reported information. Considering that a typical wireless device is resource-constrained, we also require that a user should be able to delegate the burden of auditing and detection to some public server to save its own resources. Our solution to the above public-auditing problem is constructed based on the homomorphic linear authenticator (HLA) cryptographic primitive [2], [3] which is basically a signature scheme widely used in cloud computing and storage server systems to provide a proof of storage from the server to entrusting clients. However, direct application of HLA does not solve our problem well, mainly because in our problem setup, there can be more than one malicious node along the route. These nodes may collude during the attack and when being asked to submit their reports. For example, a packet and its associated HLA signature may be dropped at an upstream malicious node, so a downstream malicious node does not receive this packet and the HLA signature from the route. However, this downstream attacker can still open a back-channel to request this information from the upstream malicious node. When being audited, the downstream malicious node can still provide valid.

II. MODULES DESCRIPTION.

1. **Service Provider:**

In this module, the service provider browses the file and sends to the particular end users via router. And also service provider can assign energy and assign distances for the nodes in router.

2. **Router:**

In this module, the router sends the file from source to destination (from service provider to end users) by selecting shortest distances between two nodes & sufficient node energy. And if node has less energy than file size then packet dropper in router drops the some packets from file and sends remaining file to the destination. And it can also do some operations like view distances, view energy, view files, view attackers, verify, refresh.

3. **Auditor:**

In this module, the auditor discovers the traffic pattern, means it stores the details of dropped packets. It contains details of in which node packets are dropped, how many no of packets dropped, from which file dropped & status of packets.

4. **Destination (End User):**

In this module, there are n no of destinations (A, B, C….). These end users only receive the file from service provider via router. While getting the file from service provider there may be chances of packets dropping, if packets are dropped then end user will gets dropped packets from point to point manager. The end users receive the file by without changing the File Contents. Users may receive particular data files within the network only.

5. **Attacker:**

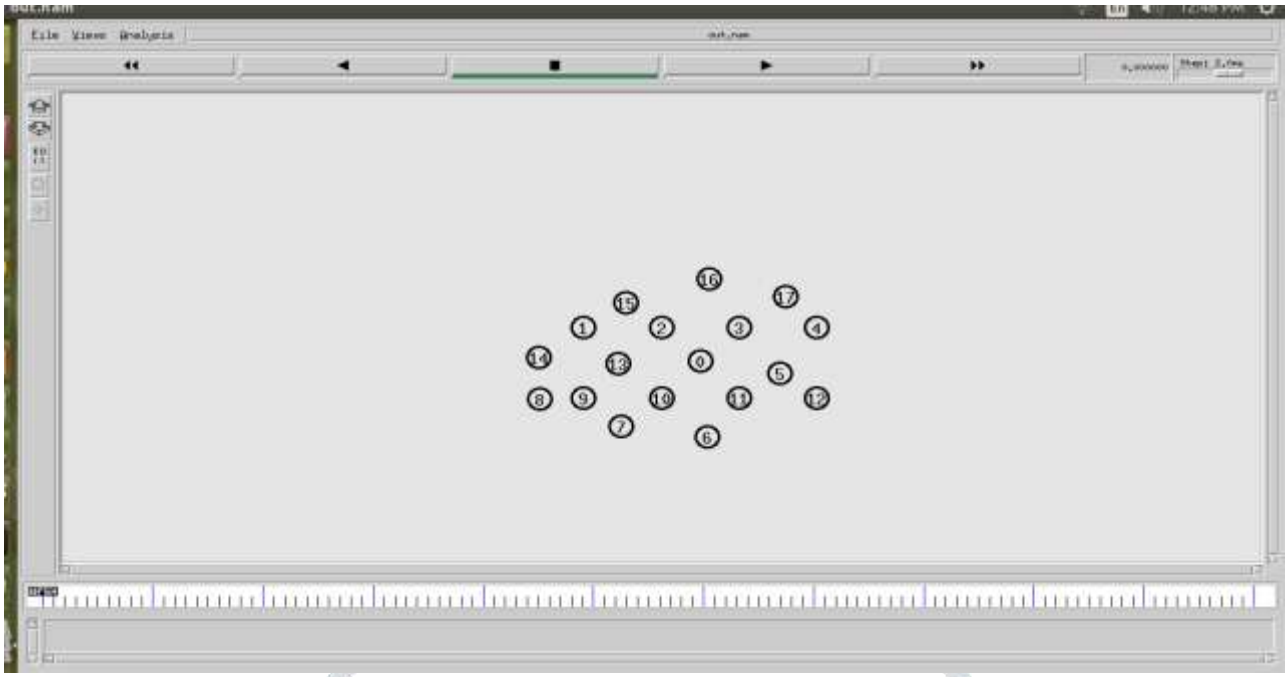
Attacker is one who makes changes the energy of particular nodes in router. And all attackers' details stored in router with their all details such as attacker Ip address, attacked node, modified energy and attacked time.

III. Ad Hoc On-Demand Distance-Vector Routing Protocol

Ad hoc on-demand distance vector (AODV) routing protocol uses an on demand approach for finding routes, that is, a route is established only when it is required by a source node for transmitting data packets. It employs destination sequence numbers to identify the most recent path. The major difference between AODV and DSR stems out from the fact that DSR uses source routing in which a data packet carries the complete path to be traversed. However, in AODV, the source node and the intermediate nodes store the next-hop information corresponding to each flow for data packet transmission. In an on demand routing protocol, the source node floods the Route Request packet in the network when a route is not available for the desired destination. It may obtain multiple routes to different destinations from a single Route Request. The major difference between AODV and other on-demand routing protocols is that it uses a destination sequence number (DestSeqNum) to determine an up-to-date path to the destination. A node updates its path information only if the DestSeqNum of the current packet received is greater than the last DestSeqNum stored at the node. A Route Request carries the source identifier (SrcID), the destination identifier (DestID), the source sequence number (SrcSeqNum), the destination sequence number (DestSeqNum), the broadcast identifier (BcastID), and the time to live (TTL) field. DestSeqNum indicates the freshness of the route that is accepted by the source. When an intermediate node receives a Route Request, it either forwards it or prepares a Route Reply if it has a valid route to the destination. The validity of a route at the intermediate node is determined by comparing the sequence number at the intermediate node with the destination sequence number in the RouteRequest packet. If a RouteRequest is received multiple times, which is indicated by the BcastID-SrcID pair, the duplicate copies are discarded. All intermediate nodes having valid routes to the destination, or the destination node itself, are allowed to sendRouteReply packets to the source. Every intermediate node, while forwarding a RouteRequest, enters the previous node address and its BcastID. A timer is used to delete this entry in case a RouteReply is not received before the timer expires. This helps in storing an active path at the intermediate node as AODV does not employ source routing of data packets. When a node receives a RouteReply packet, information about the previous node from which the packet was received is also stored in order to forward the data packet to this next node as the next hop toward the destination.

IV. RESULTS

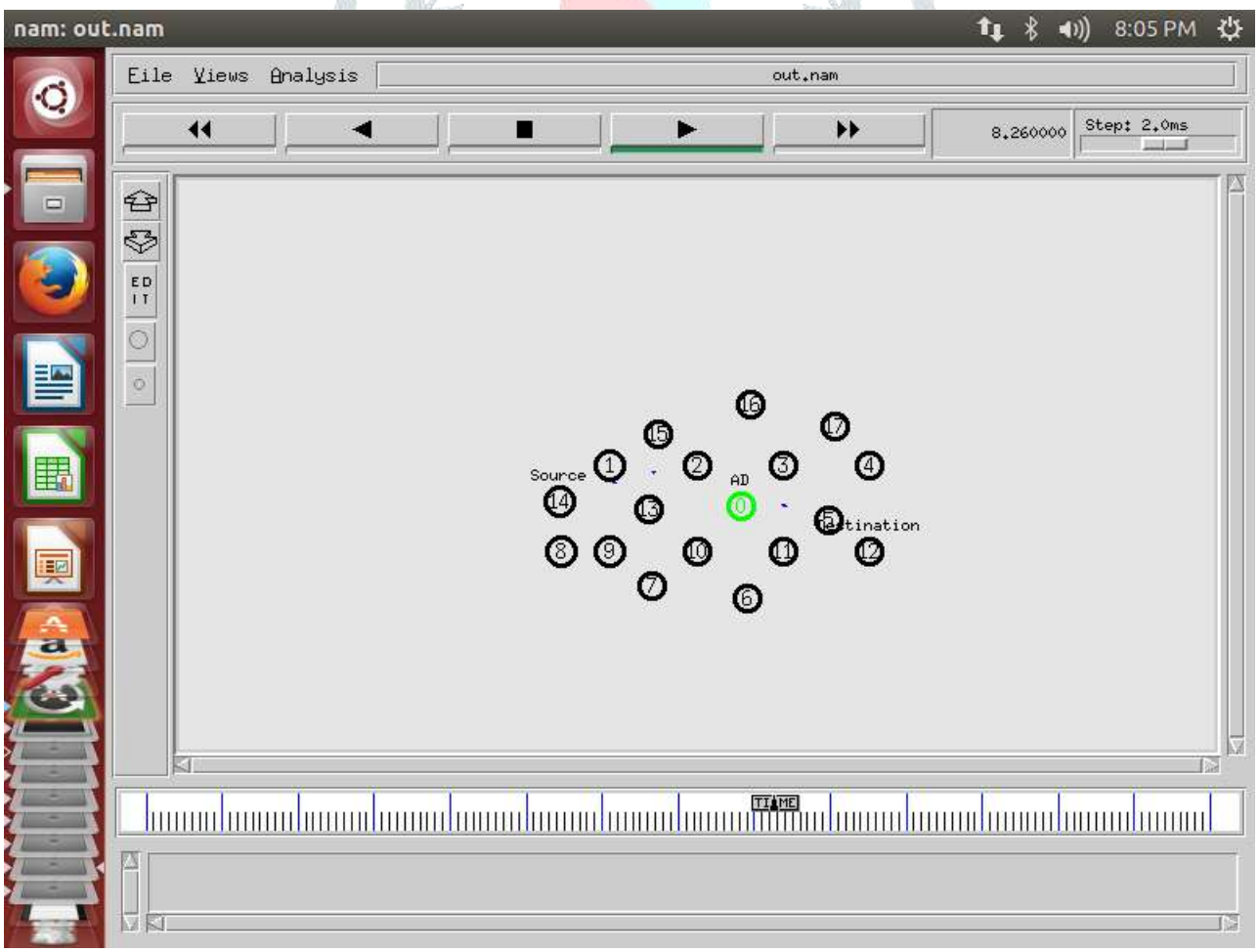
Finally in the project of detection of packet dropping attacks in wireless ad-hoc networks we found the step by step procedure for finding the true source of packet loss. The comparison between proposed and existing system has been explained in a detailed manner throw graphs.



Nodes Creation:

Path from Source to Destination:

It shows a path that exists between the source and destination node. Here the path includes the nodes 14, 2,5,12. This passes the information from source to destination on the path. The next thing after creating a node is to show the source, destination and auditor nodes. Source node is the initializing process. This consists of some basic information. Auditor node consists of all networking nodes information and also compares its present information with that of other nodes information. Destination node receives overall information from the source node.



V. THE PROCESS OF TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of

exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

a. Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

b. Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

c. Functional Test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/ Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

d. System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

e. White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

f. Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot ―seel into it. The test provides inputs and responds to outputs without considering how the software works

VI. DESIGN OF TEST CASES AND SCENARIOS

Test Approach

Testing can be done in two ways

1. Bottom up approach
2. Top down approach

Bottom Up Approach:

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

Top Down Approach:

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

Validation

It checks whether an item is being inherent the right way .It checks whether the right item is being created. Thus, so as to guarantee that the given programming is revise in all viewpoints, then it needs to fulfil the conformation and acceptance prepare effectively. Verification and Validation activities for developing software include:

- Formal Technical Review(FTR)
- Performance Monitoring
  - Feasibility Study
  - Algorithm Analysis
  - Documentation Review

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Features to be tested

Verify that the entries are of the correct format**.** No duplicate entries should be allowed**.** All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Acceptance Testing

After the system test has corrected all or most defects, the system will be delivered to the user or customer for acceptance testing. Acceptance testing is basically done by the user or customer although other stakeholders may be involved as well. The goal of acceptance testing is to establish confidence in the system. Acceptance testing is most often focused on a validation type testing. Acceptance testing may occur at more than just a single level.

The types of acceptance testing are:

The User Acceptance test: focuses mainly on the functionality thereby validating the fitness-for-use of the system by the business user. The user acceptance test is performed by the users and application managers.

The Operational Acceptance test: also known as Production acceptance test validates whether the system meets the requirements for operation. In most of the organization the operational acceptance test is performed by the system administration before the system is released. The operational acceptance test may include testing of backup/restore, disaster recovery, maintenance tasks and periodic check of security vulnerabilities.

Contract Acceptance testing: It is performed against the contract's acceptance criteria for producing custom developed software. Acceptance should be formally defined when the contract is agreed.

Compliance acceptance testing: It is also known as regulation acceptance testing is performed against the regulations.

VII. CONCLUSION

In this paper, we showed that compared with conventional detection algorithms that utilize only the distribution of the number of lost packets, exploiting the correlation between lost packets significantly improves the accuracy in detecting malicious packet drops. We showed that compared with conventional detection algorithms that utilize only the distribution of the number of lost packets, exploiting the correlation between lost packets significantly improves the accuracy in detecting malicious packet drops. Such improvement is especially visible when the number of maliciously dropped packets is comparable with those caused by link errors. To correctly calculate the correlation between lost packets, it is critical to acquire truthful packet-loss information at individual nodes. We developed an HLA-based public auditing architecture that ensures truthful packet-loss reporting by individual nodes. This architecture is collusion proof, requires relatively high computational capacity at the source node, but incurs low communication and storage overheads over the route. The implementation and optimization of the proposed mechanism under various particular protocols will be considered in our future studies.

VIII. REFERENCES

[1] J. N. Arauz, ―802.11 Markov channel modeling,‖ Ph.D. dissertation, School Inform. Sci., Univ. Pittsburgh, Pittsburgh, PA, USA, 2004.

[2] C. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, ―Provable data possession at untrusted stores,‖ in Proc. ACM Conf. Comput. and Commun. Secur., Oct. 2007, pp. 598–610. Fig. 11. Detection accuracy of block-based algorithms. 826 IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. 14, NO. 4, APRIL 2015.

[3] G. Ateniese, S. Kamara, and J. Katz, ―Proofs of storage from homomorphic identification protocols,‖ in Proc. Int. Conf. Theory Appl. Cryptol. Inf. Security, 2009, pp. 319–333.

[4] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H. Rubens, ―ODSBR: An on-demand secure byzantine resilient routing protocol for wireless ad hoc networks,‖ ACM Trans. Inform. Syst. Security, vol. 10, no. 4, pp. 1–35, 2008.

[5] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H. Rubens, ―ODSBR: An on-demand secure byzantine resilient routing protocol for wireless ad hoc networks,‖ ACM Trans. Inf. Syst. Secur., vol. 10, no. 4, pp. 11–35, 2008.

[6] K. Balakrishnan, J. Deng, and P. K. Varshney, ―TWOACK: Preventing selfishness in mobile ad hoc networks,‖ in Proc. IEEE Wireless Commun. Netw. Conf., 2005, pp. 2137–2142.

[7] D. Boneh, B. Lynn, and H. Shacham, ―Short signatures from the weil pairing,‖ J. Cryptol., vol. 17, no. 4, pp. 297–319, Sep. 2004.

[8] S. Buchegger and J. Y. L. Boudec, ―Performance analysis of the confidant protocol (cooperation of nodes: Fairness in dynamic adhoc networks),‖ in Proc. 3rd ACM Int. Symp. Mobile Ad Hoc Netw. Comput. Conf., 2002, pp. 226–236.

[9] L. Buttyan and J. P. Hubaux, ―Stimulating cooperation in selforganizing mobile ad hoc networks,‖ ACM/Kluwer Mobile Netw. Appl., vol. 8, no. 5, pp. 579–592, Oct. 2003.

[10] J. Crowcroft, R. Gibbens, F. Kelly, and S. Ostring, ―Modelling incentives for collaboration in mobile ad hoc networks,‖ presented at the First Workshop Modeling Optimization Mobile, Ad Hoc Wireless Netw., Sophia Antipolis, France, 2003.

[11] J. Eriksson, M. Faloutsos, and S. Krishnamurthy, ―Routing amid colluding attackers,‖ in Proc. IEEE Int. Conf. Netw. Protocols, 2007, pp. 184–193.

[12] W. Galuba, P. Papadimitratos, M. Poturalski, K. Aberer, Z. Despotovic, and W. Kellerer, ―Castor: Scalable secure routing for ad hoc networks,‖ in Proc. IEEE INFOCOM, Mar. 2010, pp. 1 –9.

[13] T. Hayajneh, P. Krishnamurthy, D. Tipper, and T. Kim, ―Detecting malicious packet dropping in the presence of collisions and channel errors in wireless ad hoc networks,‖ in Proc. IEEE Int. Conf. Commun., 2009, pp. 1062–1067.

[14] Q. He, D. Wu, and P. Khosla, ―Sori: A secure and objective reputation- based incentive scheme for ad hoc networks,‖ in Proc. IEEE Wireless Commun. Netw. Conf., 2004, pp. 825–830.

[15] D. B. Johnson, D. A. Maltz, and J. Broch, ―DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks,‖ in Ad Hoc Networking. Reading, MA, USA: Addison-Wesley, 2001, ch. 5, pp. 139–172.

[16] W. Kozma Jr. and L. Lazos. Dealing with liars: misbehavior identification via Renyi-Ulam games. In *Proceedings of the International ICST Conference on Security and Privacy in Communication Networks (SecureComm)*, 2009.

[17] W. Kozma Jr. and L. Lazos. REAct: resource-efficient accountability for node misbehavior in ad hoc networks based on random audits. In *Proceedings of the ACM Conference on Wireless Network Security (WiSec)*, 2009.

[18] K. Liu, J. Deng, P. Varshney, and K. Balakrishnan. An acknowledgement-based approach for the detection of routing Misbehaviour in MANETs. *IEEE Transactions on Mobile Computing*, 6(5):536–550, May 2006.

[19] Y. Liu and Y. R. Yang. Reputation propagation and agreement in mobile ad-hoc networks. In *Proceedings of the IEEE WCNC Conference*, pages 1510–1515, 2003.

[20] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the ACM MobiCom Conference*, pages 255–265, 2000.

[21] G. Noubir and G. Lin. Low-power DoS attacks in data wireless lans and countermeasures. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(3):29–30, July 2003.

[22] V. N. Padmanabhan and D. R. Simon. Secure traceroute to detect faulty or malicious routing. In *Proceedings of the ACM SIGCOMM Conference*, 2003.

[23] P. Papadimitratos and Z. Haas. Secure message transmission in mobile ad hoc networks. *Ad Hoc Networks*, 1(1):193–209, 2003.

[24] A. Proano and L. Lazos. Selective jamming attacks in wireless networks. In *Proceedings of the IEEE ICC Conference*, pages 1–6, 2010.

[25] A. Proano and L. Lazos. Packet-hiding methods for preventing selective jamming attacks. *IEEE Transactions on Dependable and Secure Computing*, 9(1):101–114, 2012.

[26] R. Rao and G. Kesidis. Detecting malicious packet dropping using statistically regular traffic patterns in multihop wireless networks that are not bandwidth limited. In *Proceedings of the IEEE GLOBECOM Conference*, 2003.

[27] H. Shacham and B. Waters. Compact proofs of retrievability. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, Dec. 2008.

[28] T. Shu, M. Krunz, and S. Liu. Secure data collection in wireless sensor networks using randomized dispersive routes. *IEEE Transactions on Mobile Computing*, 9(7):941–954, 2010.

[29] T. Shu, S. Liu, and M. Krunz. Secure data collection in wireless sensor networks using randomized dispersive routes. In *Proceedings of theIEEE INFOCOM Conference*, pages 2846–2850, 2009.

[30] C. Wang, Q. Wang, K. Ren, and W. Lou. Privacy-preserving public auditing for data storage security in cloud computing. In *Proceedings of the IEEE INFOCOM Conference*, Mar. 2010