On Securing MANET Routing Protocol Against Control Packet Dropping

Djamel Djenouri[1], Othmane Mahmoudi[1], Mohamed Bouamama[1], David Llewellyn-Jones [2], and Madjid Merabti[2].
1: Basic software Laboratory, CERIST, Algiers, Algeria.
{ddjenouri@mail.cerist.dz, o.mahmoudi@gmail.com, m.bouamama@gmail.com}
2: School of Computing and Mathematical Sciences, John Mooress University, Liverpool, UK.
{D.Llewellyn-Jones, M.Merabti@ljmu.ac.uk}

Abstract

In this manuscript we deal with securing routing protocols of mobile ad hoc networks (MANETs) against packet dropping misbehavior. More specifically, we propose a solution to protect control packets of reactive source routing protocols against. Most current proposals focus on data packets. Nonetheless, dropping control packets may be beneficial for selfish nodes and malicious ones as well. For example, simply by dropping RREQ (Route Request) packets a selfish node could exclude itself from routes and thereby avoid receiving data packets to forward. Similarly, a malicious could drop RERR (Route Error) packets to keep the use of failed routes, potentially resulting in a denial of service. Our solution could be intergraded with any source routing protocol. For the implementation in this work, we have chosen one of the most secure protocols, namely ENDAIRA. We assess our solution by an extensive simulation study.

1 Introduction

Secure routing in MANET is a topic that attracts more and more attention amongst researchers. When dealing with the selfish misbehavior or the packet dropping attack (Byzantine black hole attack), most of the current sophisticated solutions focus on data packets. These solutions are not directly applicable to control packets. The number of control packets is too low compared with data packets, thus the solutions that rely on a long experience before making a judgment (such as the Bayesian-based methods [3, 10]) are inappropriate, and a more realistic threshold is required. Further, some control packets are broadcast (RREQ), which might require a separate monitoring solution. In this paper we focus on these issues, and propose a comprehensive solution to monitor the forwarding of control packets, judge the monitored nodes, and isolate the detected misbehaving nodes. Regarding the monitoring, we propose separate solutions for directed and broadcast packets. For the first kind, we suggest the use of the two-hop ACK approach [8], and we propose a promiscuous mode based approach for the broadcast packets. As for the judgment, we propose a redemption method allowing nodes that are observed to forward packets to be redeemed. Finally, we propose to use a witness based isolation method to isolate the suspicious nodes. The remainder of this paper is organized as follows: The next section sketches related work, followed by an overview of our solution and its integration with the routing protocol ENDAIRA [1]. The forth section will present the simulation study, and the last one will conclude the paper and sketch the perspectives.

2 Related Work

Many secure routing protocols have been recently proposed for MANET. They aim at preventing the establishment of falsified routes. SAR [18] is a general proposal that can be implemented with a reactive routing protocol. It defines the trust degree that should be associated with each node, and ensures that a node is prevented from handling a RREQ (Route Request) unless it provides the required level. This way, data packets will be sent only through trusted nodes, with respect to the defined level. SAODV [18] is an implementation of SAR on AODV. One of the difficulties of this approach is the definition of the trust level. Further, assuming that nodes showing the required trust level are genuine is not always correct. SRP [17] is another secure routing protocol, based on DSR [7]. It prevents spoofing attacks, but it is vulnerable to the wormhole attack [4]. We also find this vulnerability in ARAN [6]. ARIADNE [14] is another DSR-based protocol that overcomes this attack. There are different implementations of this latter protocol; the first one is based on TESLA, the second uses MACs (Message Authentication Codes), and the most sophisticated uses digital signatures. However, it has been illustrated that this protocol is vulnerable to some fabrication attacks, which cause the construction of nonexistent routes [4]. To mitigate this attack, ENDAIRA [1] has been proposed, which is very similar to the last version of

ARIADNE. Its idea is simply to sign RREP (route reply) packets instead of RREQ ones. Note that all these secure routing protocols do not handle packet dropping misbehavior, hence they are vulnerable to the black hole attack, and to the selfish behavior.

The watchdog [15] is the first solution dealing with the packet dropping problem. Its principle is that each node in the source route monitors its successor using the promiscuous mode. For this purpose, a source routing protocol should be used. This basic solution has the advantage of not requiring any overhead as long as nodes behave well, and it could be applied both to data and control packets. Nevertheless, it is inappropriate when using the power control technique, employed by some new power-aware routing protocols following the watchdog's proposal, such as [13, 9]. Moreover, it does not deal with the isolation step. When a misbehaving node is detected, packets will be sent around it, but no measures will be taken against it, which does not prevent nodes from misbehaving. CORE [16] and CONFIDANT [3] are among the solutions that mitigate this problem, by defining some reputation and punishment strategies. But these solutions rely on the watchdog technique in their monitor component, thus inherit all its monitoring drawbacks. Moreover, they require periodic exchange of reputation information, which is costly and unnecessary so long as nodes behave well. Another monitoring approach is the employment of a kind of ACK packets known as two-hop ACK [2, 8]. An interesting optimization of this approach is the random asking strategy [12]. This latter has been used in [10]. All these ACK-based solutions focuss on data packets, and are not directly applicable to control packets. In this paper we treat this problem.

3 Solution Overview

We propose a general solution to monitor, detect, and isolate control packet droppers. We deal with both directed (unicast) and broadcast packets. For the monitoring we propose different approaches for each kind of packets. Regarding the directed packets we suggest the use of the two-hop ACK approach [8]. As the number of these packets is too low compared to data ones, the random optimization approach [12] is not efficient. The two-hop ACK is not applicable to broadcast packets, as it becomes too much costly with this kind of packets. Therefore, we propose a promiscuous-based solution to monitor control packets. Finally, We propose a redemption strategy for judgment and a reputation-based approach for isolation, applicable to directed packets as well as broadcast ones. However, the optimal values of thresholds used in judgment and isolation may change according to the kind of packets, as illustrated in the next section.

3.1 Directed Packets

The approach we suggest to use to monitor the forwarding of directed routing control packets (RREP, RERR) needs to be implemented with a source routing protocol. Each node A monitors its successor B in the source route and checks whether this latter forwards to C each packet it provides, such that C is B's successor in the source route and A could be either the source or any intermediate node. This process is repeated on each couple of hops until reaching the final destination. The solution uses a special kind of feedbacks called *two-hop ACK* [8], that travel two hops. Node C acknowledges packets sent from A by sending this latter via B a two-hop ACK. To ensure authentication of two-hop ACK packets an asymmetric cryptography-based strategy is used. Node A generates a random number and encrypts it with C's public key (PK), then appends it in the packet's header. When C receives the packet it retrieves the number, decrypts it using its secret key (SK), encrypts it using A's PK, and puts it in a two-hop ACK it sends back to A via B. In the first hop (C,B) the ACK is not transmitted in a separate packet, but piggybacked to the ordinary MAC ACK. This inclusion and emloyment of the MAC ACK reduces the number of two-hop ACK packets as much as half compared with a separate transmission on each hop. When A receives the ACK it decrypts the random number and checks whether it matches with the one it has generated, in order to validate B's forwarding regarding the appropriate packet. However, if B does not forward the packet A will not receive the two-hop ACK, and it will be able to detect this dropping after a timeout. This strategy requires a key distribution mechanisms enabling a security association between each pair of nodes. To ensure this distribution, a mechanism like the chain of trust [5] can be used. Note that the same keys could be employed for other security purposes at the other layers. As soon as the monitor node detects that the number of packets dropped by the monitored node exceeds a defined threshold, it considers this latter as misbehaving and proceeds to its isolation. More discussions of this threshold will be provided later.

Contrary to the watchdog, largely used by the current detective solutions, this approach functions well regardless the power control employment [8]. In [12], this solution has been improved by the random asking strategy, in which a monitor node does not continuously ask ACKs but it does so randomly with a coefficient that depends on the behavior of the monitored node. This strategy is efficient with data packets, and decreases considerably the overhead. However, it is inappropriate with control packets, whose number is too low, and with which we should be more severe. Remember that dropping RREPs (respectively RREQs) prevents a selfish node from being included in routes, while dropping RERRs allows a malicious node to launch a DoS

attack by preventing the destruction of broken routes. Also, note that the overhead is not an important issue for this kind of packets, since their number is low.

3.2 Broadcast Packets

For RREQs packets (which are broadcast), each node monitors every RREQ it forwards or launches as a source. The monitoring starts from the reception of the RREQ (or its launch if the node is the source) and ends after a timeout from its retransmission. For each RREQ, the transmitter monitors all its neighbors. It should either receive (or overhear) the RREQ or a RREP from every neighbor, except the node from which it received the RREQ if the node is not the source. If no one of these packets is received from a neighbor B, then the monitor notices a packet dropping for B. When a node observes that another node B drops more than the configured threshold number of packets it judges B as misbehaving, and tries to isolate it as we will see later.

3.3 Redemption

To get over false detections that may occur due to nodes mobility and channel conditions, we propose a redemption strategy for both kinds of packets. The aim is to allow a well-behaving node improving its reputation and tolerance threshold after it has been observed to drop packets due to mobility or collisions. This can be achieved by decreasing the number of packets considered dropped each time it is perceived to correctly forward packets. The pace of decreasing is not inevitably 1, but should be < 1 to prevent nodes from abusing this redemption. That is, forwarding one packet does not decrease the number of packets considered dropped by one. If the pace is m/n (such that $m, n \in \mathbb{N}, m < n$), then forwarding n packets decreases the number by m. More investigations into this parameter (redemption pace) will be performed in the next section.

3.4 Isolation

After judging a node as misbehaving, the detector attempts to isolate it. Isolating a misbehaving node means: i) do not route packets through it, to avoid losing them, and ii) do not forward packets for it, to punish it. A node A that judges some other node B as misbehaving should not punish it unilaterally, but must ensure that this will be done by all nodes. This is because when A unilaterally punishes B, the others could consider A as misbehaving when they realize that it does not forward packets for B. In social life, a person that accuses another must show proof. One possible way to prove the accusation is to get witnesses against the accused person.

Similarly, to isolate a detected node we suggest the use of a testimony-based protocol, already used with data packets [10]. Upon a detection, the detector informs nodes in its neighborhood about the dropper (the accused), and asks for witnesses by broadcasting a WREQ (Witness REQuest) packet. It also puts the detected node ID in a special set we call *a suspicious set*. Each node receiving the WREQ investigates the issue as follows:

3.4.1 Directed packets

The receiver of WREQ immediately sends a *signed* WREP (Witness REPly) packet to the accuser if its suspicious set includes the accused node (denoted by B). Otherwise, if it has not enough experience with the accused node, and if B is its neighbor then it asks the successor of this latter whether it has received packets forwarded from it, by sending an ACREQ (ACcusation REQuest) packet, using a route that does not include B. But first, in order to avoid false accusations, the investigator should ensure that the accuser has really sent a packet to B to be forwarded to the appropriate successor. One possible way to do this is to check whether such a packet has been recently overheard, using the promiscuous mode. The node also should check whether B has sent the accuser an ACK *just after* overhearing the data, to ensure that the former has really received the packet and that the latter is not impressing it. If B's successor has not recently received any packet *forwarded* from B, it sends a *signed* ACREP (ACcusation REPly) packet to the investigator, then this latter testifies for the accusation and sends the accuser a signed WREP (Witness REPly) packet.

3.4.2 Broadcast packets (RREQ)

In this case the node, if it is a neighbor of B, merely checks whether it has recently received (respectively overheard) either any RREQ *forwarded* from this node, or a RREP originated from it. To do this, each node keeps the RREQs and RREPs it receives in a buffer for a short time. If neither RREQ nor RREP have been received then it testifies for the accusation and sends the accuser a signed WREP (Witness REPly) packet. But it must first ensure that the accuser node has really recently sent out a RREQ, by checking in its buffer.

When the detector collects k validation from its neighbors, with at least one provided by direct experience (without asking the successor of B), it broadcasts in the network an accusation packet (AC) containing signatures of all the validating nodes. The requirement of at least one direct witness aims at mitigating wrong accusations caused by false

testimonies [10]. Each node receiving such a valid accusation isolates the guilty. Otherwise, if the detector fails to collect k validation then it does not punish the detected node, but keeps it in the suspicious set and could avoid sending its own packets through it. The parameter k will be investigated in the simulation study.

3.5 Implementation

As mentioned earlier, we implemented our protocol with ENDARA [1], that is based on DSR. Like all the reactive protocols, this latter includes two steps; route discovery and route maintenance. The route discovery is launched using request packets (RREQ), and consequently a route will be established using a reply packet (RREP), while the route maintenance is ensured by employing RERR packets. This basic protocol does not provide any security primitive. Attackers could create false routes, destroy valid ones, or perform any kind of attacks [11]. The most secure version of ENDAIRA proposes to apply digital signatures on RREP and RERR in each hop. And thus ensures a high security level to the routes. We integrated our protocol to this secure one, in order to enhance its security and make it immune to packet dropping misbehavior.

4 Simulation Assessment

4.1 Simulation Setup

To evaluate our solution we made an extensive simulation study using GloMoSim [19]. We simulated two kinds of packet dropping. RREQ dropping, which represents the selfish misbehavior and allows to evaluate our solution for broadcast packets, and RERR dropping that represents a malicious behavior aiming a DoS attack, which allows to evaluate the solution with respect to broadcast packets.

We simulated a network of 50 nodes, moving in an area of $1500x1000m^2$ according to the random-waypoint model [19], during 30 minutes. Each node has a power range of $250m$. We change the nodes' speed from $0m/s$ to $4m/s$, and for each value of the mobility we made the measurements for three different configurations of misbehaving: i) low misbehaving rate with 5 misbehaving nodes, ii) medium rate with 12 misbehaving nodes, iii) and finally high rate in which 20 nodes misbehave. For each configuration we used 5 seeds, resulting in no less than 2000 scenarios. The curves presented hereafter represent the averaged values for those configurations.

The simulation study has been divided into two steps. The first one establishes the best value of the intersect parameters of our protocol. Secondly, using these values we compare our protocol vs. DSR and the basic ENDAIRA, regarding both the efficiency and the cost.

4.2 Metric of Comparison

4.2.1 True Isolation Rate

The true isolation rate (TIR), or true positives, represents the efficiency on packet droppers *isolation*. It is the average rate of true isolation computed as follows:

$$TIR = \sum_{i=1, m_i \neq 0}^{n} \frac{ti_i / m_i}{k} \quad (1)$$

ti_i: is the true isolation of node i, i.e the number of misbehaving nodes monitored and detected by node i, then isolated in the network.
m_i: the number of misbehaving nodes monitored by node i.
n: the number of nodes.
k: the number of nodes that have monitored misbehaving nodes (whose $m_i \neq 0$).

4.2.2 False Isolation Rate

This metric (FIR) is very similar to the previous one. It is the average rate of false isolations, given by the following formula:

$$FIR = \sum_{i=1, m'_i \neq 0}^{n} \frac{fi_i / m'_i}{k'} \quad (2)$$

Where fi_i is the false isolations of node i, viz the number of well-behaved nodes monitored and wrongly detected by node i and isolated, m'_i is the number of well-behaving nodes monitored by node i, and finally k' is number of nodes that have monitored well-behaving nodes (whose $m'_i \neq 0$).

These metrics are involved in the two steps of the simulation. However the next ones are used in the second step, as they illustrate the cost engendered by our solution.

4.2.3 End-to-end delay

We define this metric as the average time separating the sending of a data packet from a source node and its arrival to the corresponding destination. Formally speaking:

$$delay = \frac{1}{nbRec} \sum_{i \in Rec} \sum_{j \in pr_i} \frac{delay_j}{nbpr_i} \quad (3)$$

Rec: is the set of *destination* nodes that received data packets. Nodes that did not receive any data packet are eliminated
$nbRec$: is the number of receiver nodes ($\parallel Rec \parallel$)
pr_i: is the set of packets received by node i as the final destination. Packets that did not arrive to their destination are eliminated.

$nbpr_i$: is the number of packets received ($\| pr_i \|$)
$delay_j$: is the transfer delay of packet j, such that:
$delay_j$ = packet j arrival time to its destination - packet j sending time by the source.

4.2.4 Energy

We define the average consumed power as:

$$average\_power = \sum_{i=1}^{n} \frac{PC_i}{n} \quad (4)$$

Such that PC_i is the power consumed by node i during the simulation, computed in GloMoSim using the NCR Wavelan radio model [19].

4.3 Best Intersect Parameters' Values

First, we investigate our redemption mechanism. We do this by simulating RREQ dropping in scenarios of our protocol (enforced ENDAIRA in this implementation) with and without redemptions, and compare the results. We chose RREQ packets because the number of these packets is high compared to the other control packets. Figures 1(a) and 1(b), representing respectively the false and true isolation ratios, shows how the redemption approach hugely reduces the false isolation, while keeping the true isolation close to the non-redemption version. This latter has unacceptable values of false isolations. In both figures, we explain the increase of isolation ratios (false and true) with the mobility by the fact that this latter causes more link breakage, which engenders more RREQ retransmissions, thus more packets to monitor. Note that the redemption version is much less affected by the mobility with respect to false isolations compared to the other one. The increase of true isolations, however, is a good result.

Now we try to find out the best values of parameters related to our protocol, namely the threshold number of packets upon which the node is accused, the number of witnesses, and the redemption pace, for both the broadcast (RREQ, tested by what we call the selfish behavior), and directed packets (RERR tested by the malicious behavior). We first start with broadcast ones.

From Figures 2(a) and 2(b) we realize that fixing this threshold to three strikes a balance between true and false isolations. The version representing this value has a very tolerable true detection, too close to the one representing the threshold value of two. This latter has high false isolations(up to more than 20%), while the others (threshold four and five) have too low true detections (lower than 50%). We note that the false isolations are largely affected by the mobility. This will be reduced by fixing the other parameters, as will be illustrated later.

	Parameter	Value
Broadcast packets	Tolerance threshold	3
	Number of witnesses	2
	Redemption pace	0.2
Directed packets	Tolerance threshold	1
	Number of witnesses	2
	Redemption pace	0.8

Table 1. Best parameters' values

Now we fix the previous threshold to 3, and try to find the best value for the witnesses number. As can be seen from figure 3(a), the version with two witnesses has good true isolation values, just a bit below the version with one witness and clearly above the others. On the other side, we remark in figure 3(b) that the values of the one-witness version is high and largely affected by the mobility, contrary to the other ones which are less affected by the mobility. Overall, the version with two witnesses is the one we consider well-balanced between true and false accusations. We made investigations in the same way to figure out the best value of the redemption pace. After this investigation we fixed this parameter to 0.2 ($2/5$). That is, for each 5 RREQ packets forwarded, 2 dropped are forgotten. Curves of this parameter are omitted due to space limitation.

Now we investigate the malicious RERR packets dropping. As depicted in figure 4(a), fixing the tolerance threshold to one gives much better true isolation rate compared to the other values. On the other hand, the difference regarding the false isolations between all the versions is minor (figure 4(b)). Further, they are less affected by the mobility. Therefor, we fix this parameter to one.

As for the number of witnesses, we can see that the version with three witnesses has low true isolations (figure 5(a)), while the version with one witness shows relatively high false isolations (figure 5(b)). We thus opt for two witnesses. The results concerning the redemption pace (whose curves are omitted due to space limitation) show the best value is 0.8.

Table 1 illustrates the best values of our protocol's parameters for both the directed (RREQ) and broadcast (RREP) packets. Investigations on RREP indicated that the best values for this kind if packets are the same as the one for RERR.

4.4 Comparison

Now we set the previous parameters to their best values, and compare our protocol (we note by ENDAIRA+) with the basic ENDAIRA and DSR. Our protocol clearly outperforms both DSR and ENDAIRA with respect to packet dropper isolation, since those latter simply do not detect such a misbehavior. Figure 6(a) shows how our protocol,

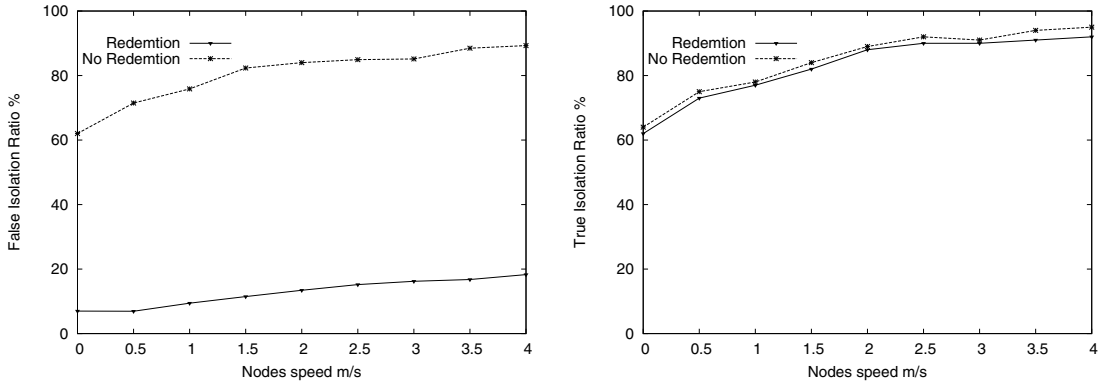


Figure 1. Redemption Mechanism

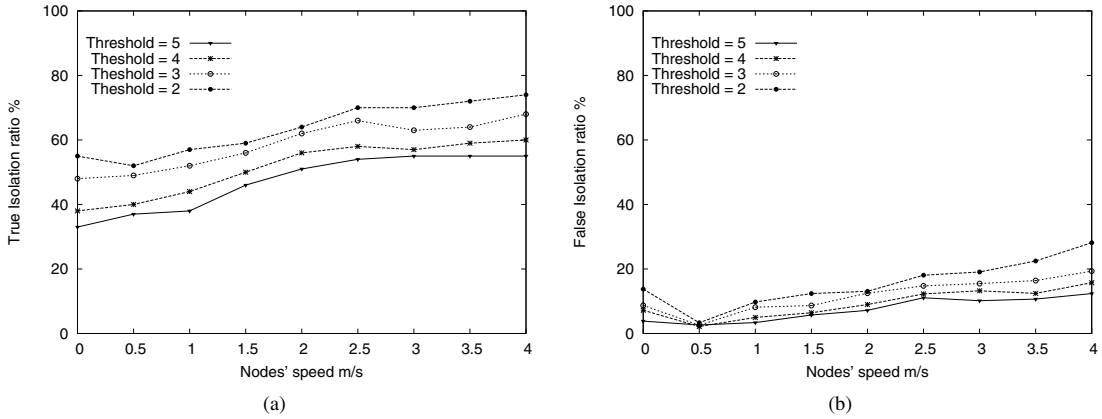


Figure 2. Number of Packets Threshold (Selfish Behavior)

has high true isolations, especially when the mobility increase. On the other hand, figure 6(b) shows that the false isolation rate has been considerably reduced when fixing optimally the parameters, and more importantly that the protocol becomes less affected with the mobility. The cost of this misbehavior detection is a small rise in both delay and power consumption, compared with ENDAIRA. For the delay, presented in figure 7(a), the big difference between DSR and the other secure protocols is basically due to cryptographic primitives (digital signatures computations on RREP packets) used by those latter. The increase with the mobility can be argued by the fact that mobility causes the launching of more route discoveries, thus more latency due to cryptography computation before sending the data packets. The most important issue here is the minor difference between ENDAIRA and our protocol. This difference is due to the monitoring procedures. Finally, we observe almost the same differences regarding energy (figure 7(b)). The difference between the protocols is due to the overhead. The small difference between ENDAIRA and our protocol indicates that the cost of the control packets added by the latter (overhead) is minor. The only difference between this figure and the previous one is the reduction of the power consumed (especially for the secure protocols) with the mobility. This is mainly due to the increase of packets lost when the mobility is risen. This was not observed for the delay because the packets lost are not used for the computation of this metric.

All these results were obtained by simulating RREP dropping. All but the same results were obtained with the RERR dropping simulation. Curves of these latter are omitted due to space limitations.

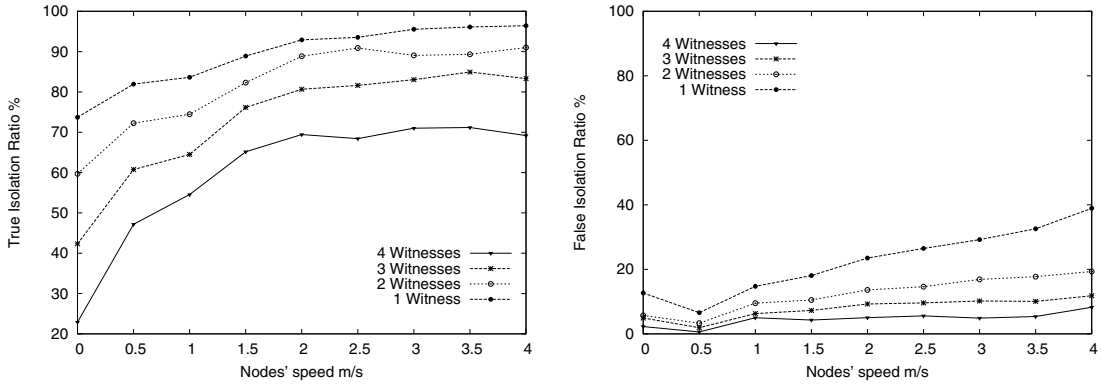


Figure 3. Number of Witnesses (Selfish Behavior)

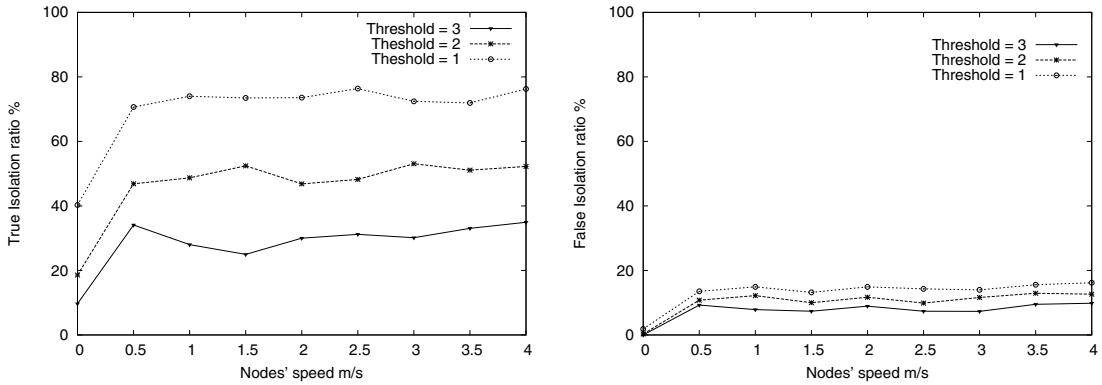


Figure 4. Number of Packets Threshold (Malicious Behavior)

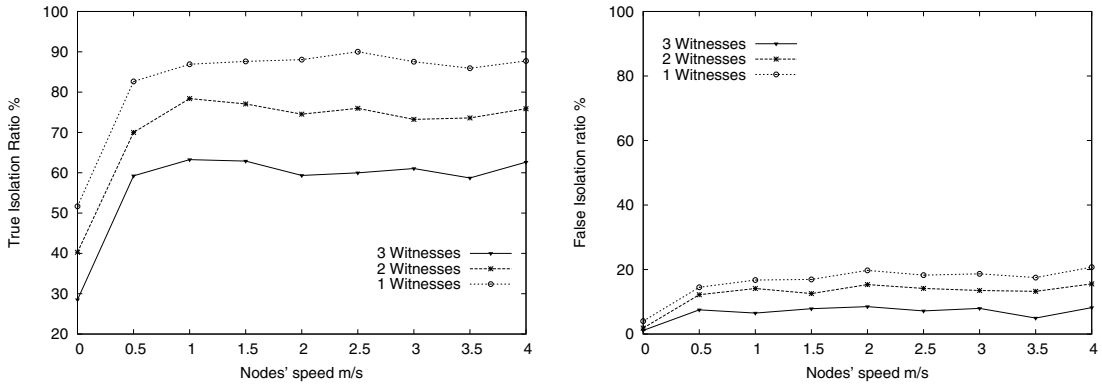
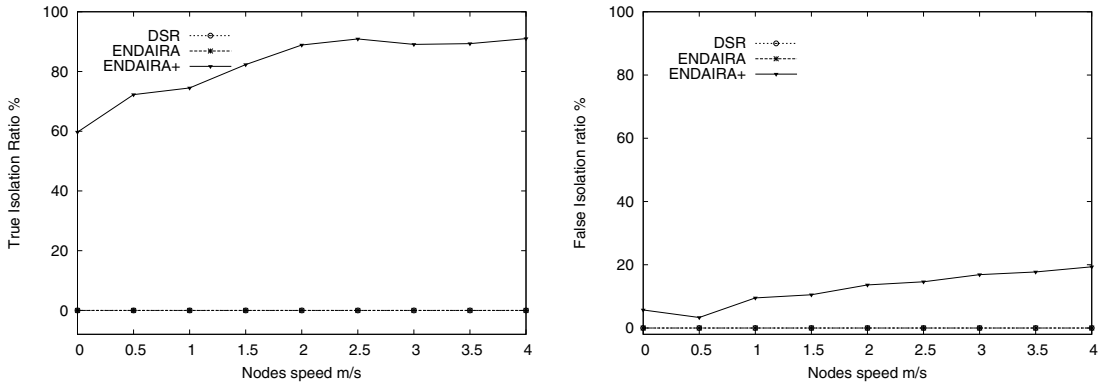
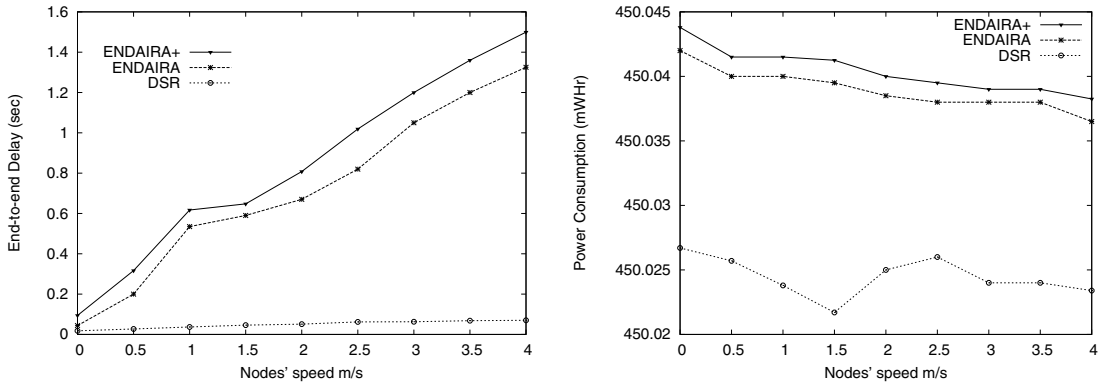


Figure 5. Number of Witnesses (Malicious Behavior)



(a) (b)

Figure 6. True and False Isolation of Protocols



(a) (b)

Figure 7. End-to-end Delay and Power Consumption of Protocols

5 Conclusion

In this paper we proposed a general solution to packet dropping misbehavior in mobile ad hoc networks. The solution allows to monitor, detect, and isolate the droppers. We implemented the solution with the secure source routing protocol ENDAIRA, and made a comprehensive simulation study to first fix the crucial parameters of our solution to optimal values, and then to compared it with the basic protocols. Usually, the MANET nodes' mobility causes degradation in efficiency of protocols, but in our case we remarked that it helps improving the true positives of our solutions. Nonetheless, we also remarked that it degrades the false positives. After setting the parameters to the optimal values, the latter metric becomes less affected by the increase of the mobility, which renders the protocol adaptable to the mobility. Compared with ENDAIRA, the cost of our protocol is really minor, in both latency and power consumption. However, there is an important difference between these secure protocols and DSR, as this latter does not ensure any security primitive. This difference is caused mainly by the employment of digital signatures, which indeed are robust but costly. Implementing our solution with another lighter secure routing protocol could represent a perspective to this work. Proposing a solution for self setting the parameters according to network configuration and conditions also represents a potential perspective.

References

[1] G. Acs, L. Buttyan, and I. Vajda. Provably secure on-demand source routing in mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(11):1533–1546, 2006.

[2] K. Balakrishnan, J. Deng, and P. K. Varshney. Twoack: preventing selfishness in mobile ad hoc networks. In *The IEEE Wireless Communications and Networking Conference(WCNC'05)*, pages 2137– 2142, New Orleans, LA, USA, March 2005.

[3] S. Buchegger and J.-Y. Le-Boudec. A robust reputation system for p2p and mobile ad-hoc networks. In *Second Workshop on the Economics of Peer-to-Peer Systems*, Harvard university, Cambridge, MA, USA, June 2004.

[4] L. Buttyan and I. Vajda. Towards provable security for ad hoc routing protocols. In *The ACM Workshop on Security in Ad Hoc and Sensor Networks SASN04*, Washington DC, October 2004.

[5] S. Capkun, L. Buttyan, and J.-P. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):52–64, January 2003.

[6] B. Dahill, B. N. Levine, E. Royer, and C. Shields. Aran: A secure routing protocol for ad hoc networks. Technical Report UMass Tech Report 02-32, University of Massachusetts, Amherst, 2002.

[7] B. David and A. David. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, volume 353, pages 153–181. Kluwer Academic, imielinskiand korth edition, 1996.

[8] D. Djenouri and N. Badache. A novel approach for selfish nodes detection in manets: Proposal and petri nets based modeling. In *The 8th IEEE International Conference on Telecommunications (ConTel'05)*, pages 569–574, Zagreb, Croatia, June 2005.

[9] D. Djenouri and N. Badache. New power-aware routing for mobile ad hoc networks. *The International Journal of Ad Hoc and Ubiquitous Computing (Inderscience Publisher)*, 1(3):126–136, 2006.

[10] D. Djenouri and N. Badache. Testimony-based isolation: New approach to overcome packet dropping attacks in manet. In *The 7th Postgraduate Symposium on Convergence of Telecommunications, Networking and Broadcasting (PgNet'06)*, pages 114–119, John Moors University, Liverpool, UK, June 2006.

[11] D. Djenouri, L. Khalladi, and N. Badache. A survey of security issues in mobile ad hoc and sensor networks. *IEEE Communications Surveys*, 7(4):2–28, 2005.

[12] D. Djenouri, N. Ouali, A. Mahmoudi, and N. Badache. Random feedbacks for selfish nodes detection in mobile ad hoc networks. In *The 5th IEEE International Workshop on IP Operations and Management, IPOM'05*, number 3751 in LNCS, pages 68–75, Barcelona, Spain, October 2005. Springer-Verlag GmbH.

[13] S. Doshi and T. Brown. Minimum energy routing schemes for a wireless ad hoc network. In *The 21st IEEE Annual Joint Conference on Computer Communications and NetworkingINFOCOM'02*, New York, USA, 2002.

[14] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne:: a secure on-demand routing protocol for ad hoc networks. In *The 8th annual international conference on Mobile computing and networking MobiCom '02*, pages 12–23. ACM Press, 2002.

[15] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *ACM Mobile Computing and Networking, MOBICOM 2000*, pages 255–65, Boston, MA, USA, 2000.

[16] P. Michiardi and R. Molva. Core: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *The 6th IFIP Communication and Multimedia Security Conference*, Portoroz, Slovenia, September 2002.

[17] P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. In *The SCS Communication Networks and Distributed Systems Modeling and Simulation Conference CNDS02*, San Antonio, Texas, 2002.

[18] S. Yi, P. Naldurg, and R. Kravets. Security-aware ad hoc routing for wireless networks. In *The ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC01)*, Long Beach, CA, October 2001.

[19] X. Zeng, R. Bagrodia, and M. Gerla. Glomosim: A library for the parallel simulation of large-scale wireless networks. In *The 12th Workshop on Parallel and distributed Simulation. PADS'98*, pages 154–161, Banff, Alberta, Canada, May 1998.