



## Detection of Packet Dropping Attacks in Wireless AdHoc Networks

<sup>1</sup>P. V. N. Rajeswari, <sup>2</sup>B. Krishnaiah, <sup>3</sup>G. Venkateswar Rao

<sup>1</sup>Asso. Prof., Dept of C.S.E., Visvodaya Engineering College, Kavali, India

<sup>2</sup>M-Tech., Student Dept of C.S.E.

<sup>3</sup>Asso. Prof., Dept of Information Technology, GITAM University, Vizag, India

---

**Abstract**—Link error and malicious packet dropping are two sources for packet losses in multi-hop wireless ad hoc network. In this paper, 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. We are especially interested in the insider-attack case, whereby malicious nodes that are part of the route exploit their knowledge of the communication context to selectively drop a small amount of packets critical to the network performance. Because the packet dropping rate in this case is comparable to the channel error rate, conventional algorithms that are based on detecting the packet loss rate cannot achieve satisfactory detection accuracy. To improve the detection accuracy, we propose to exploit the correlations between lost packets. Furthermore, to ensure truthful calculation of these correlations, we develop a homomorphism linear authenticator (HLA) based public auditing architecture that allows the detector to verify the truthfulness of the packet loss information reported by nodes. This construction is privacy preserving, collusion proof, and incurs low communication and storage overheads.

**Index Terms**--Packet dropping, secure routing, attack detection, homomorphism linear signature, auditing.

---

### I. INTRODUCTION

IN a multi-hop wireless network, nodes cooperate in relaying/routing traffic. An adversary can exploit this cooperative nature to launch attacks. For example, the adversary may first pretend to be a cooperative node in the route discovery process. Once being included in a route, the adversary starts dropping packets. In the most severe form, the malicious node simply stops forwarding every packet received from upstream nodes, completely disrupting the path between the source and the destination. Eventually, such a severe denial-of-service (DoS) attack can paralyze the network by partitioning its topology. 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 amount that is deemed highly critical to the operation of the network. In this paper, we are interested in combating such an insider attack. In particular, we are interested in the 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 (or hop) 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 (e.g., fading, noise, and interference, a.k.a., link errors), 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. In this paper, 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 (equivalently, different patterns of packet losses). 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.

## II. PROBLEM STATEMENT

Under the system and adversary models defined above, we address the problem of identifying the nodes on PSD that drop packets maliciously. We require the detection to be performed by a public auditor that does not have knowledge of the secrets held by the nodes on PSD. When a malicious node is identified, the auditor should be able to construct a publicly verifiable proof of the misbehaviour of that node. The construction of such a proof should be privacy preserving, i.e., it does not reveal the original information that is transmitted on PSD. In addition, the detection mechanism should incur low communication and storage overheads, so that it can be applied to a wide variety of wireless networks.

## III. PROPOSED DETECTION SCHEME

### 3.1 Overview

The proposed mechanism is based on detecting the correlations between the lost packets over each hop of the path. The basic idea is to model the packet loss process of a hop as a random process alternating between 0 (loss) and 1 (no loss). Specifically, consider that a sequence of  $M$  packets that are transmitted consecutively over a wireless channel. By observing whether the transmissions are successful or not, the receiver of the hop obtains a bitmap  $(a_1, \dots, a_M)$ , where  $a_j \in \{0, 1\}$  for packets  $j = 1; \dots, M$ . The correlation of the lost packet is calculated as the auto-correlation function of this bitmap. To correctly calculate the correlation between lost packets, it is critical to enforce a truthful packet-loss bitmap report by each node. We use HLA cryptographic primitive for this purpose. The basic idea of our method is as follows. An HLA scheme allows the source, which has knowledge of the HLA secret key, to generate HLA signatures  $s_1; \dots, s_M$  for  $M$  independent messages  $r_1; \dots, r_M$ , respectively. The source sends out the  $r_i$ 's and  $s_i$ 's along the route. The HLA signatures are made in such a way that they can be used as the basis to construct a valid HLA signature for any arbitrary linear combination of the messages,  $\sum_{i=1}^M c_i r_i$ , without the use of the HLA secret key, where  $c_i$ 's are randomly chosen coefficients. A valid HLA signature for  $\sum_{i=1}^M c_i r_i$  can be constructed by a node that does not have knowledge of the secret HLA key if and only if the node has full knowledge of  $s_1; \dots, s_M$ . So, if a node with no knowledge of the HLA secret key provides a valid signature for  $\sum_{i=1}^M c_i r_i$ , it implies that this node must have received all the signatures  $s_1; \dots, s_M$ . Our construction ensures that  $s_i$  and  $r_i$  are sent together along the route, so that knowledge of  $s_1; \dots, s_M$  also proves that the node must have received  $r_1; \dots, r_M$ . Our detection architecture consists of four phases: setup, packet transmission, audit, and detection. We elaborate on these phases in the next section.

### 3.2 Scheme Details

#### 3.2.1 Setup Phase

This phase takes place right after route  $P_{SD}$  is established, but before any data packets are transmitted over the route. In this phase,  $S$  decides on a symmetric-key crypto-system ( $encrypt_{key}$ ,  $decrypt_{key}$ ) and  $K$  symmetric keys  $key_1, \dots, key_K$ , where  $encrypt_{key}$  and  $decrypt_{key}$  are the keyed encryption and decryption functions, respectively.  $S$  securely distributes  $decrypt_{key_j}$  and a symmetric key  $key_j$  to node  $n_j$  on  $P_{SD}$ , for  $j=1, \dots, K$ . Key distribution may be based on the public-key crypto-system such as RSA. Besides symmetric key distribution,  $S$  also needs to set up its HLA keys. Let  $e : G \times G \rightarrow G_T$  be a computable bilinear map with multiplicative cyclic group  $G$  and support  $Z_p$ , where  $p$  is the prime order of  $G$ , i.e., for all  $\alpha, \beta \in G$  and  $q_1, q_2 \in Z_p$ ,  $e(\alpha^{q_1}, \beta^{q_2}) = e(\alpha, \beta)^{q_1 q_2}$ . Let  $g$  be a generator of  $G$ .  $H_2(\cdot)$  is a secure map-to-point hash function:  $\{0, 1\}^* \rightarrow G$ , which maps strings uniformly to  $G$ .  $S$  chooses a random number  $x \in Z_p$  and computes  $v = g^x$ . Let  $u$  be another generator of  $G$ . The secret HLA key is  $sk = x$  and the public.

#### 3.2.2 Packet Transmission Phase

After completing the setup phase,  $S$  enters the packet transmission phase.  $S$  transmits packets to  $P_{SD}$  according to the following steps. Before sending out a packet  $P_i$ , where  $i$  is a sequence number that uniquely identifies  $P_i$ ,  $S$  computes  $r_i = H_1(P_i)$  and generates the HLA signatures of  $r_i$  for node  $n_j$ , as follows:

$$s_{ji} = [H_2(i||j)u^x]^x, \text{ for } j=1, \dots, K,$$

where  $||$  denotes concatenation. These signatures are then sent together with  $P_i$  to the route by using a one-way chained encryption that prevents an upstream node from deciphering the signatures intended for downstream nodes. More specifically, after getting  $s_{ji}$  for  $j=1, \dots, K$ ,  $S$  iteratively computes the following:

$$c \text{ HLA key is a tuple } pk = (v, g, u). \tilde{s}_{Ki} = \text{encrypt}_{key_K}(S_{Ki}),$$

$$\mathfrak{K}_i = \tilde{s}_{Ki} || \text{MAC}_{key_K}(\tilde{s}_{Ki}),$$

$$\tilde{s}_{K-1i} = \text{encrypt}_{key_{K-1}}(S_{K-1i} || \mathfrak{K}_i),$$

$$\mathfrak{K}_{K-1i} = \tilde{s}_{K-1i} || \text{MAC}_{key_{K-1}}(\tilde{s}_{K-1i}),$$

$$\tilde{s}_{ji} = \text{encrypt}_{key_j}(s_{ji} || \mathfrak{K}_{j+1i}),$$

$$\mathfrak{K}_{ji} = \tilde{s}_{ji} || \text{MAC}_{key_j}(\tilde{s}_{ji}),$$

$$\tilde{s}_{1i} = \text{encrypt}_{key_1}(s_{1i} || \mathfrak{K}_{2i}),$$

$$\mathfrak{K}_{1i} = \tilde{s}_{1i} || \text{MAC}_{key_1}(\tilde{s}_{1i}),$$

where the message authentication code (MAC) in each stage  $j$  is computed according to the hash function  $H_{key_j}^{MAC}$ . After getting  $t_{1i}$ ,  $S$  puts  $P_i || \mathfrak{K}_{1i}$  into one packet and sends it to node  $n_1$ . When node  $n_1$  receives the packet from  $S$ , it extracts  $P_i$ ,  $\tilde{s}_{1i}$ , and  $\text{MAC}_{key_1}(\tilde{s}_{1i})$  from the received packet. Then,  $n_1$  verifies the integrity of  $\tilde{s}_{1i}$  by testing the following equality:

$$\text{MAC}_{\text{key}1}(s_{1i}) = H_{\text{key}1}^{\text{MAC}}(s_{1i}).$$

If the test is true, then  $n_1$  decrypts  $s_{1i}$  as follows:

$$\text{decrypt}_{\text{key}}(s_{1i}) = s_{1i} \parallel \mathcal{L}_{2i}$$

Then,  $n_1$  extracts  $s_{1i}$  and  $\mathcal{L}_{2i}$  from the decrypted text. It stores  $r_i = H_1(P_i)$  and  $s_{1i}$  in its proof-of-reception database for future use. This database is maintained at every node on  $P_{SD}$ . It can be considered as a FIFO queue of size  $M$ , which records the reception status for the most recent  $M$  packets sent by  $S$ . Finally,  $n_1$  assembles  $P_i \parallel \mathcal{L}_{2i}$  into one packet and relays this packet to node  $n_2$ . In case the test in (5) fails,  $n_1$  marks the loss of  $P_i$  in its proof-of-reception database and does not relay the packet to  $n_2$ .

The above process is repeated at every intermediate node  $n_j, j = 1; \dots; K$ . As a result, node  $n_j$  obtains  $r_i$  and its HLA signature  $s_{ji}$  for every packet  $P_i$  that the node has received, and it relays  $P_i \parallel \mathcal{L}_{j+1i}$  to the next hop on the route. The last hop, i.e., node  $n_K$ , only forwards  $P_i$  to the destination  $D$ . Here we consider the verification of the integrity of  $P_i$  as an orthogonal problem to that of verifying the tag  $\mathcal{L}_{ji}$ . If the verification of  $P_i$  fails, node  $n_1$  should also stop forwarding the packet and should mark it accordingly in its proof-of-reception database.

### 3.2.3 Audit Phase

This phase is triggered when the public auditor  $A_d$  receives an ADR message from  $S$ . The ADR message includes the id of the nodes on  $P_{SD}$ , ordered in the downstream direction, i.e.,  $n_1; \dots; n_K$ ,  $S$ 's HLA public key information  $p_k = (v, g, u)$  the sequence numbers of the most recent  $M$  packets sent by  $S$ , and the sequence numbers of the subset of these  $M$  packets that were received by  $D$ . Recall that we assume the information sent by  $S$  and  $D$  is truthful, because detecting attacks is in their interest.  $A_d$  conducts the auditing process as follows.

$A_d$  submits a random challenge vector  $c_j = (c_{j1}, \dots, c_{jM})$  to node  $n_j, j = 1; \dots; K$ , where the elements  $c_{ji}$ 's are randomly chosen from  $Z_p$ . Without loss of generality, let the sequence number of the packets recorded in the current proof-of-reception database be  $P_1; \dots; P_M$ , with  $P_M$  being the most recent packet sent by  $S$ . Based on the information in this database, node  $n_j$  generates a packet-reception bitmap vector  $b_j = (b_{j1}, \dots, b_{jM})$  where  $b_{ji} = 1$  if  $P_i$  has been received by  $n_j$ , and  $b_{ji} = 0$  otherwise. Node  $n_j$  then calculates the linear combination  $r^{(j)} = \sum_{i=1, b_{ji} \neq 0}^M c_{ji} r_i$  and the HLA signature for the combination as follows:

$$s^{(j)} = \prod_{i=1, b_{ji} \neq 0}^{c_{ji}} s_{ji}$$

Node  $n_j$  submits vector  $b_j, r^{(j)}$ , and  $s^{(j)}$  to  $A_d$ , as proof of the packets it has received.  $A_d$  checks the validity of  $r^{(j)}$ , and  $s^{(j)}$  by testing the following equality:

$$e(s^{(j)}, g) = e(\prod_{i=1, b_{ji} \neq 0}^{c_{ji}} s_{ji} u^{r^{(j)}}(v))$$

If the equality holds, then  $A_d$  accepts that node  $n_j$  received the packets as reflected in vector  $b_j$ . Otherwise,  $A_d$  rejects vector  $b_j$  and judges that not all packets claimed in vector  $b_j$  are actually received by  $n_j$ , so  $n_j$  is a malicious node.

### 3.2.4 Detection Phase

The public auditor  $A_d$  enters the detection phase after receiving and auditing the reply to its challenge from all nodes on  $PSD$ . The main tasks of  $A_d$  in this phase include the following: detecting any overstatement of packet loss at each node, constructing a packet-loss bitmap for each hop, calculating the autocorrelation function for the packet loss on each hop, and deciding whether malicious behaviour is present. More specifically,  $A_d$  performs these tasks as follows. Given the packet-reception bitmap at each node, vectors  $b_1; \dots; b_K$ ,  $A_d$  first checks the consistency of the bitmaps for any possible overstatement of packet losses. Clearly, if there is no overstatement of packet loss, then the set of packets received at node  $j+1$  should be a subset of the packets received at node  $j$ , for  $j=1, \dots, K$ .

1. Because a normal node always truthfully reports its packet reception, the packet-reception bitmap of a malicious node that overstates its packet loss must contradict with the bitmap of a normal downstream node. Note that there is always at least one normal downstream node, i.e., the destination  $D$ . So  $A_d$  only needs to sequentially scan vector  $b_j$ 's and the report from  $D$  to identify nodes that are overstating their packet losses. After checking for the consistency of vector  $b_j$ 's,  $A_d$  starts constructing the per-hop packet-loss bitmap vector  $m_j$  from vector  $b_{j-1}$  and vector  $b_j$ . This is done sequentially, starting from the first hop from  $S$ . In each step, only packets that are lost in the current hop will be accounted for in  $m_j$ . The packets that were not received by the upstream node will be marked as "not lost" for the underlying hop. Denoting the "lost" packet by 0 and "not lost" by 1, vector  $m_j$  can be easily constructed by conducting a bit-wise complement-XOR operation of vector  $b_{j-1}$  and vector  $b_j$ . For example, consider the following simple case with three intermediate nodes (four hops) on the route and with  $M = 10$ . Suppose that vector  $b_1 = (0; 1; 1; 1; 1; 1; 1; 1; 0; 1)$ , vector  $b_2 = (0; 1; 1; 1; 1; 1; 1; 1; 0; 1)$ , vector  $b_3 = (0; 1; 0; 1; 1; 0; 1; 1; 0; 1)$ , and the destination  $D$  reports that vector  $b_D = (0; 1; 0; 1; 1; 0; 1; 1; 0; 1)$ . Then the per-hop packet-loss bitmaps are given by vector  $m_1 = (0; 1; 1; 1; 1; 1; 1; 1; 0; 1)$ , vector  $m_2 = (1; 1; 1; 1; 1; 1; 1; 1; 1; 1)$ , vector  $m_3 = (1; 1; 0; 1; 1; 0; 1; 1; 1; 1)$ , and vector  $m_4 = (1; 1; 1; 1; 1; 1; 1; 1; 1; 1)$ . The auditor calculates the autocorrelation function  $g_j$  for each sequence vector  $m_j = (m_{j1}; \dots; m_{jM}), j=1; \dots; K$ , as follows:

$$g_j(i) = \sum_{k=1}^{M-i} (m_{jk} m_{j,k+1})$$

$$M - i$$

$$\text{for } i=0, \dots, M-1; j=1, \dots, K.$$

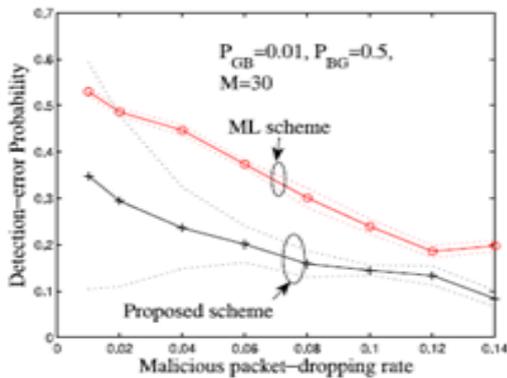
The auditor then calculates the relative difference between  $g_j$  and the ACF of the wireless channel  $f_c$  as follows:

$$\mathcal{E}_j = \sum_{i=0}^{M-1} \frac{g_j(i) - f_c(i)}{f_c(i)}$$

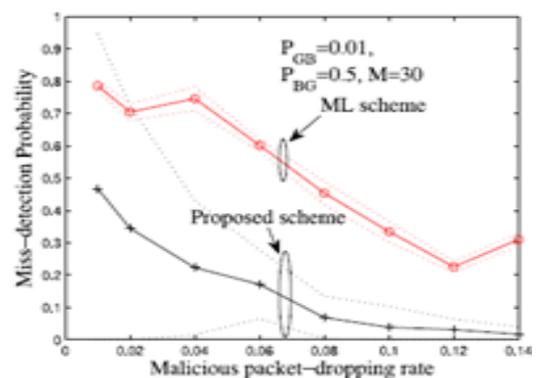
$$f_c(i)$$

The relative difference  $j$  is then used as the decision statistic to decide whether or not the packet loss over the  $j^{\text{th}}$  hop is caused by malicious drops. In particular, if  $\epsilon_j \geq \epsilon_{\text{th}}$ , where  $\epsilon_{\text{th}}$  is an error threshold, then  $A_d$  decides that there is malicious packet drop over the hop. In this case, both ends of the hop will be considered as suspects, i.e., either the transmitter did not send out the packet or the receiver chose to ignore the received packet.  $S$  may choose to exclude both nodes from future packet transmissions, or alternatively, apply a more extensive investigation to refine its detection. For example, this can be done by combining the neighbour overhearing techniques [12] used in the reputation system. By fusing the testimony from the neighbours of these two nodes,  $A_d$  can pin-point the specific node that dropped the packet. Once being detected, the malicious node will be marked and excluded from the route to mitigate its damage. *Public verifiability.* After each detection,  $A_d$  is required to publish the information it received from involved nodes, i.e.,  $b^{\rightarrow j}$ ,  $r^{(j)}$ ,  $s^{(j)}$ , for  $j \in P_{SD}$ , so that a node can verify all calculation has been performed correctly. Note that no knowledge of the HLA secret key  $x$  is required in the verification process. At the same time, because  $A_d$  has no knowledge of  $x$ , there is no way for it to forge a valid HLA signature for  $r^{(j)}$ . In other words,  $A_d$  cannot claim a misbehaving node to be a normal one.

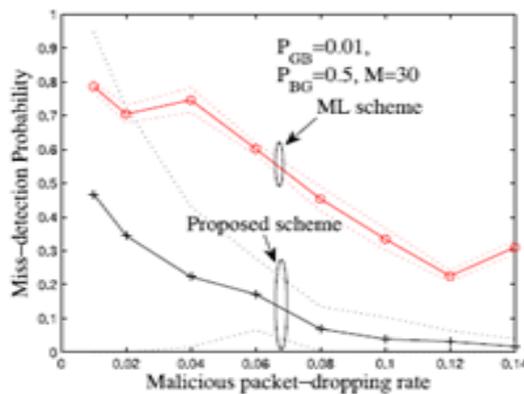
#### IV. RESULTS



(a) Overall detection-error probability



(b) Miss-detection probability



(b) Miss-detection probability

#### V. CONCLUSIONS

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. 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. Some open issues remain to be explored in our future work. First, the proposed mechanisms are limited to static or quasi-static wireless ad hoc networks. Frequent changes on topology and link characteristics have not been considered extension to highly mobile environment can be studied in our future work

#### 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.
- [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

- [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.