Journal of Theoretical and Applied Information Technology
30th June 2014. Vol. 64 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ENHANCED SCENARIO-BASED EXPERIMENTS TO OVERCOME BYZANTINE ATTACKS IN MANET

[1]G.MURUGABOOPATHI, [2]D.RAJALAKSHMI, [3]R.JAYANTHAN

[1]Department of Information Technology, Vel Tech Multitech Dr. Rangarajan Dr. Sakunthala Engineering College
[2]Research Scholar Department of Computer Science and Engineering, Vel Tech Multitech Dr. Rangarajan Dr. Sakunthala Engineering College
[3]Research Scholar Department of Computer Science and Engineering, SCSVMV University

E-Mail: [1]gmurugaboopathi@gmail.com, [2]rajisacet@gmail.com, [3]jayanthtomail@gmail.com

ABSTRACT

MANETs are unplanned, self-configuring network composed of mobile nodes that utilize mesh networking principles for inter-connectivity. MANETs are IP networks made up of a collection of wireless and mobile nodes communicating via radio links that can temporarily form a network whenever they coexist in the same neighbourhood without any fixed infrastructure such as base stations for mobile switching and no centralized administration. Generally the nodes have a limited transmission range, such that each node seeks some assistance of its neighbouring nodes to forward packets. The main problem in MANETs is the security which is because of the open nature and no fixed topology of the MANET environment. One of the primary goals of designing secure routing protocols is to prevent a compromised node from disrupting the route discovery and maintenance mechanisms. However, this added security comes at the cost of performance. This paper evaluates the performance of a secure routing protocol to overcome byzantine attacks using scenario-based experiments.

Keywords *- Mobility, Ad-hoc, Security, Routing, Cryptography, MANET*

1. INTRODUCTION

In MANETs world, devices such as laptops, PCs, cellular phones, appliances with ad hoc communication capability link together on the fly to create a network. With each node acting as a router and dynamically changing topology the availability is not always guaranteed in MANETs. It is also not guaranteed that the path between two nodes would be free of malicious nodes. The wireless links between nodes are highly susceptible to link attacks (passive eavesdropping, active interfering, etc). Under the framework of network security, security solutions can be provided in different layers of the Open Systems Interconnection (OSI) network model. It is critical to provide secure routing protocols in the network layer [1] [2][14], [16], [18] that can defend the most possible attacks against routing, which are data and routing information tampering. Ad hoc routing protocols [8], must be integrated into authentication architectures, such as Public Key Infrastructure (PKI) and Certificate Authority (CA), relevant to all security related issues such as authenticity, confidentiality, integrity and non-repudiation services.

This paper proposes a novel attack detection and defence algorithm to solve the Byzantine attack problems in MANETs.

2. RELATED WORK

On the basis of security mechanisms, the MANET routing protocols [1], [22], can be classified into two that is embedding the mechanism of security with the existing routing protocols and the other mechanism which will detect and defend such security related attacks.

In the first category, the common approach is to secure the popular on-demand routing protocols, that includes Adhoc on Demand Distance Vector Routing (AODV) [5], [9],[10] Destination Sequenced Distance Vector (DSDV), and Dynamic Source Routing (DSR) [7], by using a security association between the source and destination nodes such as pair wise secret keys and end-to-end authentication. The resulting secure protocols include Secure AODV (SAODV), Ariadne, Secure Efficient Ad hoc Distance (SEAD), and Authenticated Routing [11] for Ad hoc Networks (ARAN) [3].

SAODV is a direct extension of AODV that uses a digital signature to sign routing messages and

Journal of Theoretical and Applied Information Technology
30th June 2014. Vol. 64 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

hash chains to secure hop counts, which is expensive for MANETs. Ariadne with Timed Efficient Stream Loss-tolerant Authentication (TESLA) [12] can be considered as an extension of DSR with added security features to prevent attackers from tampering routing information and some other types of attacks such as DOS. TESLA is an efficient broadcast scheme for authentication, but it requires time synchronization to some extent among the nodes in a MANET. SEAD is based on DSDV [3],[6], and uses one-way hash chains to authenticate hop counts and sequence numbers of routing messages. The security mechanism in SEAD can be TESLA or the shared secret keys between each pair of nodes. ARAN provides end-to-end authentication, node authentication, message integrity, and non repudiation services [4].

During route discovery, each routing message is signed by a source node and then broadcast to others. An intermediate node that forwards the message removes its previous hop's certificate and signature, and then attaches its own certificate and signature. During route setup, each message is similarly signed twice and uni-cast back to its source. Because of the use of a double signature, ARAN can defend from the most common attacks. As an authenticated routing protocol [11] [17], ARAN can work with both AODV and DSR.

3. DYNAMIC KEY MANAGEMENT SCHEME

The most possible attacks in the network layer are Tampering of routing information and the data attack [25],[26].To prevent most of the external attacks, the technology used is simple link layer encryption and authentication [20]. The novel concept is proposed in which every node shares a unique symmetric key with the source if it needs to transmit data.

A. Key Management Scheme

The basic two key management approaches are the Public and secret key-based schemes. The first one uses a pair of public/private keys and an asymmetric algorithm. The second scheme is a symmetric key shared by two nodes with a secret key for data integrity.

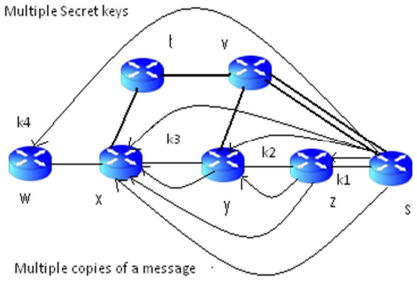

Fig. 1. Demonstration Of Message And Route Redundancy

There are various methods to set up the shared keys:

1) Bootstrap the shared keys from a PKI, which might be a strong assumption for MANETs; 2) use a key distribution center, that has a shared key with each node, to build up a shared key between two nodes by using the Kerberos protocol; or 3) embed the shared keys in each node during its initialization before deployment. In this paper, we assume that each node has a unique ID or address and an initial pair of public/private keys, which can be embedded into each node at the initialization of the network, or created by a self-organized public key management system [21] [27].

First we define a network, as shown in Fig. 1, and then describe a framework of dynamic key management. Let G = (V ;E) be a network whose vertices in V are nodes and whose edges in E are direct wireless links among nodes. We define for each node x the set $N_1(x)$, which contains the vertices in the network G that are hop-1 or direct neighbors of x, which is given as

$$N_{1(x)} = \{y: (x: y) \in E \text{ and } y = x\} \quad (1)$$

We can also define the hop-2 neighbors of a node as follows. For each node x, $\mathbf{N_2(x)}$ contains the vertices in the network G that are hop-2 neighbors of x, which include neither vertices in $\mathbf{N_1(x)}$ nor x itself, i.e.,

$$N_{2(x)} = \{z: (y: z) \in E \text{ and } y \in N_1(x), z = x\} \quad (2)$$

Similarly, we can define the hop-*n* neighbors of $x[N_n(x)]$ in terms of $N_{n-1}(x)$ if the flooding path from the source to destination has *n* links.

Initially, a node *x* has a public key $\boldsymbol{K_{x,pub}}$ that is distributed to $\boldsymbol{N_1(x)}$ by using PKI or CA. Similarly, a node *y* has public key $\boldsymbol{K_{y,pub}}$ distributed to $N_1(y)$. Thus, for example, if $y \in N_1(x)$ and $x \in N1(y)$, i.e., *x* and *y* are hop-1 neighbors, then *x* can authenticate

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

y by issuing a certificate (which is a proof of *y*'s ID and public key with *x*'s signature) that is signed by *x* with *x*'s private key. Those who hold *x*'s public key can now read the certificate and trust the binding of *y* and its public key. Based on the available certificate and key information, two hop-1 neighboring nodes can easily establish a secret key between them by using methods such as a three-way handshake.

B. Key Distribution and Node Authentication

We define the notations as follows:

s denotes the sender node;
r denotes the receiver node;
$K_{s,pub}$ and $K_{s,pri}$ denote the public and private keys of node *s*, respectively;
E(*m,K*) denotes the public key encryption algorithm [20] with a key *K* on message *m*, where *m* = *M* + {*IDf* } + *SN*, and *M* is the original message;
IDf denotes the ID of *f*, which is the node that forwards the message *m*;
SN is the sequence number of the message; and
h(*m* + *k*) denotes the keyed hash algorithm with a key *k* on message *m*,
where + denotes the concatenation of strings.

It can be seen that any node that handles the message has to append its ID for non-repudiation service. The ID is protected together with the forwarded message.

Whenever there is a need to initiate a route discovery process, the node creates a pair-wise shared keys with intermediate nodes, hop by hop, until it reaches the destination. First, it picks random number num and it signs num with a private key by using a public key algorithm like RSA [21]. Then, the route discovery message is protected by a keyed hash MAC algorithm such as MD5. Finally, the hash value and signature can now be attached to the route discovery message and sent out to its neighbours. The complete Route Request (RREQ) packet sent by the node can be summarized as

$$m + h(m + num) + E(num, k_{s,pri}) \quad (3)$$

Those who are *s*'s neighbors , have its public key are able to verify the signature and thus decrypt the key in the message.

Suppose that *z* $N_1(s)$ is one of *s*'s hop-1 neighbors. When there is a need for *s* to initiate a route discovery process, it picks a key k_1 at random, which serves as the shared secret key between *s* and *z*. *s* encrypts the key *k*1 by using its neighbor's public key $K_{z,pub.}$ Then, it encrypts the above encrypted key by using its own private key $K_{s,pri}$. This result is to use as signature to discover the route message, which is protected by a keyed hash MAC algorithm such as MD5. The complete procedure is called Keyed MD5. The complete RREQ sent by *s* can be summarized as

$$m_q + h(m_q + k_1) + E(E(k_1 k_{z,pub}), k_{s,pri}), for Z \in N_1(s) \quad (4)$$

where $\boldsymbol{m_q}$ stands for the message used in RREQ. This way, only the node that has *z*'s private key can read the key *k*1, the receiving node is also assured that the key and message come from *s*, and finally the integrity of message *m* can be verified by the receiving node after it decrypts the key. Then, *z* sends back *s* a route reply (RREP) packet in a similar format

$$m_p + h(m_p + k_1) + E(E(k_{1,} k_{s,pub}), k_{z,pri}), for Z \in N_1(s) \quad (5)$$

where $\boldsymbol{m_p}$ stands for the message used in RREP. By decrypting the message and comparing the key, *s* can authenticate *z* and distribute a shared key to *z*. Similarly, *s* establishes a shared key with each of its hop-1 neighbors.

Suppose that *y* $\in N_1(z)$. *z* can also similarly find out its hop-1 neighbors and also establishes a shared key with each of them. For *s* to send messages to its hop-2 neighbors, i.e., $N_2(s)$, for example, *y*, *s* requests *z* to forward the message to *y*. In *z*'s handshaking with *y*, *z* can pick *s*'s public key instead of a random key and send it to *y*. This way, *s*'s public key can be delivered to its hop-2 neighbors. Also*s* can obtain the public keys of its hop-2 neighbors in the same fashion.

By checking the acknowledgement message back from *y* via *z*, *s* can find out all of its hop-2 neighbors $N_2(s)$. Therefore, *s* can send a message to $r \in N_2(s)$ via $z \in N_1(s)$ in the following format:

$m_2 + h(m_2 + k_1)$, k_1 = shared key between *s* and *y*

where

$$\mathbf{m_2} = \mathbf{m + h(m + k_2) + E(E(k_2, k_{r,pub}), k_{s,pri})\ for\ r \in N_2(s)} \quad (6)$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

where $\boldsymbol{k_2}$ is the shared key between *s* and its hop-2 neighbor *r*. Similarly, by using the double hash and signature operations, the shared key between *s* and its hop-*n* neighbors, i.e., k_n, is created by *s* and distributed to $\boldsymbol{N_n(s)}$where $n = 2, 3, \ldots$

In the above key distribution process, the same message *m* has been sent to the destination multiple times and protected by different secret keys at each time and this is known as *message redundancy*.

C. Route Discovery and Attack Detection

In this section, we extend our algorithm in detecting collusion to Byzantine attacks [23] [28], in which two or more nodes collude to drop, fabricate, modify, or misroute packets, and these nodes are consecutively located on a path.

1) Detection of a Single Malicious Node: The basic mechanism for a node to detect misbehaving nodes is to compare the different copies of the same message it has received via different routes or at different times. The nodes along a route can be identified by checking the aggregated node IDs that are attached to the message. When a message comes from different intermediate nodes, it has to be decrypted by using different shared keys.

To be more specie, we assume that *z* (in Fig. 1) is a compromised node during the route discovery phase, although it is initially authenticated. Clearly, *z* could not tamper the message from *s* to *y* because the message is protected with a key between *s* and *y*. Of course, *z* may simply drop the message when it needs to forward the message to *y*. However, there are at least two copies of the same message *y* expects to receive. By comparing these copies from other neighbors, *y* is still able to detect that *z* is faulty or compromised. Similarly, *y* can also detect other internal attacks, such as message fabrication caused by *z*. Therefore, the attacks initiated by a single inside node can be detected [17].

2) Detection of Two Colluding Nodes: A more challenging case is the Byzantine attack [23]. In our design of key management schemes, a source has directly established a shared key with each of its hop-*n* neighbors.

Suppose that both *z* and *y* are compromised and colluding. In addition, *s* shares a hop-1 key with *z* (i.e., $k_{1,sz}$), a hop-2 key with *y* (i.e., $k_{2,sy}$), and a hop-3 key with *x* (i.e., $k_{3,sx}$). During route discovery, *x* may receive three copies of a message *m* from *s* and via different intermediate nodes *y* and *z*, respectively, in the following formats:

$$
\begin{aligned}
C_1 &= m + h(m + k_{3,sx}) \\
C_2 &= m + h\big(m + k_{2,sy}\big) + h\big(m + h\big(m + k_{2,sy}\big) \\
&\qquad + k_{1,yx}\big) \\
C_3 &= m + h\big(m + K_{1,sz}\big) \\
&\qquad + h\big(m + h\big(m + k_{1,sz}\big) + K_{1,zy}\big) \\
&\qquad + h\big(m + h\big(m + K_{1,sz}\big) + K_{1,yx}\big)
\end{aligned}
\tag{7}
$$

Suppose that z and y are two colluding nodes. It is assumed that the source and destination, i.e., s and x, are trusted via some external mechanisms. Note that each copy of the message is verified by an intermediate node along a route. As a single node, z cannot tamper the message without being detected.

Let us assume that z has modified the message but y does not tell during its forwarding. After having received the three copies from s, x finds the discrepancies among C_1, C_2, and C_3. Note that C_1 directly comes from s and thus can be trusted; y cannot change the message without being detected, and thus, C_2 must match C_1. Therefore, C_3 has been modified, and x finds that there may be some compromised or faulty nodes among the nodes that forward the message, e.g., z and/or y. It can be seen from C_3 that z may modify the message and then forwards it to y, who also gets a copy of the message directly from s as seen in C_2. If y reports the discrepancies of the two copies, then z must be a compromised node. Otherwise, both y and x are compromised and colluding nodes, although y does not change the message.

3) Detection of More Colluding Nodes: In the same manner, for the case of three colluding nodes consecutively located on a route, their collusion can also be detected if there exists at least four copies of the message that arrives at the receiver.

In general, it is not only to detect the collusion of n compromised nodes that are consecutively located on a route but also to identify these nodes, a receiver must have at least n + 1 copies of the same message, and one of the copies is more trusted than the others. The copies can either go through different routes or be protected by the shared keys in different segments of a route.

Thus the internal attacks originated through the single compromised node and the Byzantine attacks can be detected without using expensive aggregated signatures, which are used to protect a route from end to end.

It is also noted that the trustworthiness of the source node can be solved only via external

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

mechanisms such as PKI, by using such mechanisms as key refreshing, rekeying, and revoking.

We also note that the redundant use of shared keys between a source and each intermediate and the destination node may result in a scalability problem. For example, if there are n nodes along a route, then the dynamic key management scheme needs to create and distribute $(n-1)^2/2$ keys to the nodes on the route. Therefore, it is not appropriate for networks with a large number of low-resource nodes.

4. SIMULATION RESULTS

In this section, an NS-2 simulator is used to investigate the performances compared with other protocols. The routing protocol [9] is denoted as SPAC(Secure Protocol Against Collision) [18] [19],

A. Network Models and Parameters

The parameters and values are given in Table 1 for the simulations used to compare SPAC to AODV. The node mobility is generated using a random way point model wherein a node starts off at a random point in the topology. The radio propagation model used is a two-ray ground reflection that accounts for a realistic physical scenario.

Table 1: Simulation Parameters For SPAC And AODV

Number of nodes	50
Topology dimensions	1000m X 1000m
Radio range	250m
Node pause times	0-40s
Maximum node speed	1-20m/s
Source- destination pairs	20
Traffic Pattern	FTP/TCP
Data payload size	32-1060 bytes/packet

The simulations are conducted on a Dell Power Edge server with two Intel Xeon processors of 2.66 GHz and 4-GB SDRAM running in a Linux OS of Fedora Core version 3.0. The encryption and decryption operations on routing packets are simulated by adding a time of T_n= 16.38 ms (based on a CPU of 2.8 GHz) to the processing time per packet per node, as shown in Section IV-D. The RSA key size is assumed to be 1024 bits. The encryption and decryption times are invoked whenever a node generates, receives, or forwards a routing packet, which increases the overhead of SPAC as compared to AODV. However, the total overhead has in fact been reduced in the presence of malicious nodes in the network, as shown in the following sections.

To include the behaviour of malicious nodes into the simulations, SPAC is implemented by modifying the AODV protocol in NS-2. We make the following assumptions on a malicious node.

1) It does not have knowledge about the public key of its hop-1 neighbors prior to the route-discovery phase. The session keys are established on demand.

2) Once the route-discovery phase is accomplished, a malicious node randomly drops both routing and data packets and selectively only drops data packets with a specific probability, i.e., the dropping ratio.

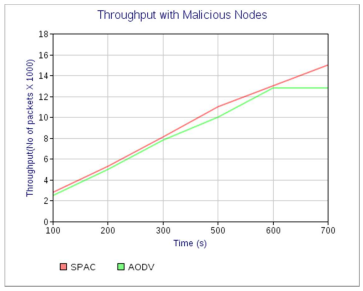

Fig. 2. Total Throughput In The Presence Of Five Malicious Nodes.

In this simulation, as a demonstration purpose, a malicious node just drops data packets. It is worthy noting that other malicious behaviors, such as false advertising, misrouting, and violating security rules, can be detected and defended in the same way as dropping. We also assume that a source–destination pair (SD pair) is trusted and cannot turn to be malicious throughout the operation of all these protocols.

B. Performance Evaluation

The performance metrics are defined as follows.

1) Total throughput: The total number of data (application) packets that have been received at time *t* by a destination node.

2) Total overhead: The total number of routing (control) packets that have been transmitted at time *t* by the nodes in the network.

3) Packet latency: The time elapsed since a data packet is transmitted to the time when it is received at the destination.

4) Packet Delivery Ratio (PDR): The ratio of the total number of data packets successfully delivered to the destination to the total number of data packets sent out by a source node. In our first scenario, simulations are conducted to examine the performance impact of adding security to routing

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

protocols [16].

Here, SPAC is compared to AODV as the implementation of SPAC is based on AODV. A malicious node randomly drops data packets and can be detected during topology discovery. The dropping ratio is in the range of 20%–50%. Each simulation is run for 700 s to collect the performance data.

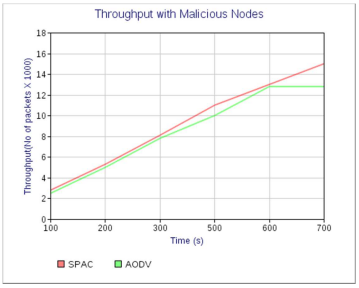

Fig. 3. Total Throughput In The Presence Of Ten Malicious Nodes.

Fig. 2 shows the total throughput of SPAC and AODV [13] in the presence of FIVE malicious nodes out of 50. The number of malicious nodes is so low that AODV continues delivering packets, although with a less amount than SPAC. This is because of the fact that the packet dropping is not serious and there are enough uncompromised nodes available to establish routes between SD pairs.

In this and other secure routing protocols the computational burden at each node is still a major issue during deployment [2], [14], [18]. It requires both analytical investigations and engineering considerations. For example, how many neighbors should a node have without degrading network performance and security? How many copies should a node receive before sending back an acknowledgement? At present, SPAC considers the link performance as a routing metric. Considering the mobility in SPAC is expected to increase the prediction accuracy and thus reduce the link breakage rate during deployment. All these problems will further be investigated in future work.

However, if the number of malicious nodes increases from 5 to 10, and to 20, as shown in Fig. 3 and 4, the number of packets delivered by AODV drastically decreases, whereas SPAC still delivers almost the same amount of data as in the previous case. The reason is that most of the routes in AODV have to go through some malicious nodes and thus result in a high packet drop or low PDR. Eventually, AODV stops delivering packets at t= 550and 500 s in the presence of 20% and 40% malicious nodes, respectively. Due to the packet drop, a connection will be timed out, and a new route discovery will be reinitiated. However, with a high probability, the new routes may again contain some malicious nodes and thus result in high data loss. On the other hand, SPAC tries to discover the paths that can go around malicious nodes. Even when the number of malicious nodes is relatively high, SPAC still discovers trustworthy routes and thus assures successful packet delivery [24].

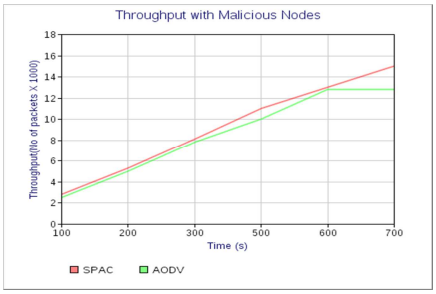

Fig. 4. Total throughput in the presence of 20 malicious nodes

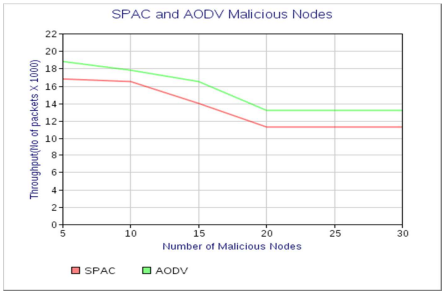

Fig. 5 Shows The Total Overhead Of SPAC and AODV

It can be seen that SPAC always has a smaller overhead than AODV in the presence of different numbers of malicious nodes. The reason is that SPAC can detect malicious nodes and thus exclude them from routing. For AODV, if a node on an established route becomes malicious and starts dropping packets, the source that waits for the acknowledgment (ACK) eventually times out. A new route discovery is initiated to re-establish the route. As the number of malicious nodes increases, AODV tends to wait and time out more often, thus delivering increasingly fewer routing and data packets. Thus, the total overhead is reduced. For SPAC, since the behavior of neighboring nodes is monitored, the malicious nodes are detected and excluded from routing. As the number of nodes that join routing becomes fewer, the total overhead is

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

reduced to a larger extent than that of AODV.

A malicious node randomly drops both routing and data packets with a dropping ratio of 80%. The maximum nodal speed varies between 1.25 and 10 m/s. Fig. 6 shows the PDR of SPAC at different speeds as compared to AODV.

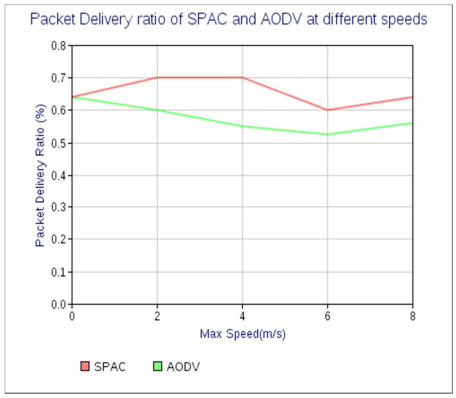

Fig. 6. Packet Delivery Ratio Of SPAC And AODV At Different Speeds.

It can be seen that SPAC always outperforms AODV in PDR, because SPAC always chooses more reliable routes by avoiding malicious nodes. At low levels of mobility, as the maximum speed increases from 1.25 to 2.5 m/s, although the increased link breakage may reduce the PDR, the SD pairs are more likely to find available nodes to forward the packets. At high levels of mobility, as the maximum speed increases from 2.5 and 10 m/s, the link breakage becomes the major cause that reduces the PDR. The SD pairs do not have adequate time to seek alternative routes due to a fixed route request timer. Therefore, the PDR decreases as the maximum speed increases.

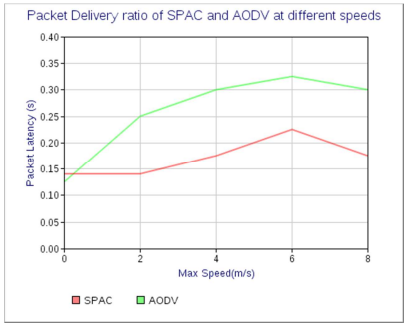

Fig. 7. Packet Latency Of SPAC And AODV At Different Speeds.

Fig. 7 shows the packet latency of SPAC at different speeds as compared with AODV. For both SPAC and AODV, as the speed increases, the information on a route in the routing table will quickly be out of date. Thus, both take more time to establish routes. However, SPAC always has lower packet latency than AODV. One reason is the use of multipath routing in SPAC. For AODV, its routing table only stores one path. It has to re-establish another route once a link is broken. For SPAC, its routing table contains multiple paths. If one fails, an alternate one will immediately be available. Further, both the packet performance and route reliability are considered in the routing metrics [15].

5. CONCLUSION

Attack detection and defence mechanism by using the route redundancy in ad hoc networks is proposed in this paper. An optimal routing algorithm by combining trustworthiness and performance of the network is also discussed. The proposed method quantitatively considers the detection of difficult internal attacks as well as the network performance.

The simulation results have shown the effectiveness of the proposed attack detection algorithm over a well known protocol AODV. The proposed attack detection and routing algorithms can be integrated into existing routing protocols for MANETs, specifically the AODV and DSR.

REFERENCES

[1] P. Papadimitratos and Z. Haas, "Securing the Internet routing infrastructure," *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 60–68, Oct. 2002.

[2] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad Hoc Netw.*, vol. 1, no. 2/3, pp. 293–315, Sep. 2003.

[3] C. Zhang, M. C. Zhou, and M. Yu, "Ad hoc network routing and security: A review," *Int. J. Commun. Syst.*, vol. 20, no. 8, pp. 909–925,Aug. 2007.

[4] J.-S. Hwu, R.-J. Chen, and Y.-B. Lin, "An efficient identity-based cryptosystem for end-to-end mobile security," *IEEE Trans.Wireless Commun.*, vol. 5, no. 9, pp. 2586–2593, Sep. 2006.

[5] C. E. Perkins and E. M. Royer, "The ad hoc on-demand distance-vector protocol," in *Ad Hoc Networking*, C. E. Perkins, Ed. Reading, MA: Addison-Wesley, 2001, ch. 6, pp. 173–220.

[6] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

(DSDV) for mobile computers," in *Proc. ACM SIGCOMM Conf. Comms. Architectures, Protocols Appl.*, 1994, pp. 234–244.

[7] D. B. Johnson, D. A. Maltz, and J. Broch, "DSR: The dynamic source routing protocol for multihop wireless ad hoc networks," in *Ad Hoc Networking*, C. E. Perkins, Ed. Reading, MA: Addison-Wesley, 2001, ch. 5, pp. 139–172.

[8] M. G. Zapata and N. Asokan, "Securing ad hoc routing protocols," in *Proc. ACM WiSe*, Atlanta, GA, Sep. 28, 2002, pp. 1–10.

[9] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure on demand routing protocol for ad hoc networks," in *Proc. 8th ACM Int. Conf. Mobile Comput. Netw.*, Sep. 2002, pp. 12–23.

[10] Y.-C. Hu, D. B. Johnson, and A. Perrig, "SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks," in *Proc. 4th IEEE Workshop Mobile Comput. Syst. Appl.*, Jun. 2002, pp. 3–13.

[11] K. Sanzgiri, D. LaFlamme, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer, "Authenticated routing for ad hoc networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 3, pp. 598–610, Mar. 2005.

[12] W. Liu, W. Lou, and Y. Fang, "An efficient quality of service routing algorithm for delay-sensitive applications," *Comput. Netw.*, vol. 47, no. 1, pp. 87–104, Jan. 2005.

[13] K. K. Leung and L.-C. Wang, "Integrated link adaptation and power control to improve error and throughput performance in broadband wireless packet networks," *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 619–629, Oct. 2002.

[14] P. Papadimitratos and Z. Haas, "Secure routing for mobile ad hoc networks," in *Proc. SCS Commun. Netw. Distrib. Syst. Model. Simul. Conf.*, Jan. 2002, pp. 27–31.

[15] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in *Proc. NDSS*, San Diego, CA, Feb. 8–9, 2001, pp. 90–100.

[16] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Rushing attacks and defense in wireless ad hoc network routing protocols," in *Proc. ACM WiSe*, Sep. 2003, pp. 30–40.

[17] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: A defense against wormhole attacks in wireless networks," in *Proc. INFOCOM*, Hong Kong, Apr. 2003, pp. 1976–1986.

[18] I. Avramopoulos, H. Kobayashi, R.Wang, and A. Krishnamurthy, "Highly secure and efficient routing," in *Proc. IEEE INFOCOM*, Hong Kong, Mar. 2004, pp. 197–208.

[19] M. T. Goodrich, "Leap-frog packet linking and diverse key distributions for improved integrity in network broadcasts," in *Proc. IEEE Symp. Security Privacy*, 2005, pp. 196–207.

[20] D. Boneh, C. Gentry, H. Shacham, and B. Lynn, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proc. Eurocrypt*, 2003,vol. 2656, p. 641.

[21] S. Capkun, L. Buttyan, and J. Hubaux, "Self-organized public-key management for mobile ad hoc networks," *IEEE Trans. Mobile Comput.*,vol. 2, no. 1, pp. 1–13, Jan.–Mar. 2003.

[22] J. Dowling, E. Curran, R. Cunningham, and V. Cahil, "Using feedback in collaborative reinforcement learning to adaptively optimize MANET routing," *IEEE Trans. Syst.,Man, Cybern. A, Syst., Humans*, vol. 35, no. 3,pp. 360–372, May 2005.

[23] B. Krishnamachari and S. Iyengar, "Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE Trans. Comput.*, vol. 53, no. 3, pp. 241–250, Mar. 2004.

[24] A. A. Pirzada and C. McDonald, "Establishing trust in pure ad-hoc networks," in *Proc. 27th ACSC*, 2004, pp. 47–54.

[25] T.-W. Kwon, C.-S. You, W.-S. Heo, Y.-K. Kang, and J.-R. Choi, "Two implementation methods of a 1024-bit RSA crypto processor based on modified Montgomery algorithm," in *Proc. IEEE ISCAS*, May 6–9, 2001,vol. 4, pp. 650–653.

[26] Z. Wang, L. Liu, and M. C. Zhou, "Space and network diversity combination for masked node collision resolution in wireless ad hoc network," *IEEE Trans. Wireless Commun.*, vol. 6, no. 2, pp. 478–485, Feb. 2007.

[27] G.Murugaboopathi N.Insozhan, S.Vinod ," Selfish Nodes Detection Using Random 2ack In MANET"s", International Journal of Emerging Science and Engineering, Vol 1 no. 4 pp.1862-1868.feb 2013

[28] Parthasarathy Vellusamy, Murugaboopathi Gurusamy, M.J.Carmel Mary Belinda," POR: Position based Opportunistic Routing for Reliable and Efficient Data Transmission in MANETs", Life Science, Vol 10 no 2, pp.1862-1868 June 2013.