A Secure Routing Scheme for Mobile Ad Hoc Networks

ILKER BASARAN
DEPARTMENT OF COMPUTER SCIENCE, UC RIVERSIDE

1. Introduction

An ad hoc network is a group of wireless mobile computers, in which nodes cooperate by forwarding packets for each other to allow them to communicate beyond direct wireless transmission range. Ad hoc networks require no centralized administration or fixed network infrastructure such as base stations or access points, and can be quickly and inexpensively set up as needed.

Nodes in an ad hoc network can communicate with each other at any time, subject to connectivity limitations. The communication in mobile ad hoc networks consists two parts, route discovery and data transmission. Both parts are subject to various numbers of attacks. I must note that secure routing protocols, which ensure the correctness of the discovered topology information, cannot by themselves ensure the secure and undisrupted delivery of transmitted data. This is because malicious nodes may behave accordingly with the route discovery and place on utilized routes, then in an unexpected time they do their harm.

So securing route discovery or data transmission alone will not help. While handling those issues, one also should be aware of DoS, replay, man-in-the-middle, resource consumption, and wormhole attacks. I think assuming centralized key distribution or a certification authority is contrary to wireless ad hoc network spirit and understanding. Because, in ad hoc wireless networks, no base stations exist and each mobile node acts as both a router and a host. But an adaptive solution to security issues in mobile ad hoc networks should consider with and without a central authority. In the case of having a central authority, distribution of keys (shared, public, private) is easier and convenient. In these cases, the broadcast authentication scheme, TESLA, will be used in detecting authenticity of route messages in an efficient way. On the other hand, if we do not have a central authority, establishing the trust between pairs is harder. The neat idea of SUCV (Statistically Unique and Cryptographically Verifiable) identifiers and addresses may be used in that situation.

Due to infrastructurless design of wireless ad hoc networks, traditional security and cryptographic mechanisms are not suitable to use. Especially when we consider computing power and battery life of these mobile devices, one should be careful about these issues while designing security schemes.

My approach here will basically be on discovering secure routes throughout the network and try to maintain this property as time progresses. There are several possible types of attack to mobile ad hoc networks, and I will try to provide scalable solutions to them.

2. Background

2.1. DSR

The design of this secure routing scheme is based on the basic operation of Dynamic Source Routing protocol. DSR is an on-demand routing protocol that is able to react quickly many changes that may occur in node connectivity. It performs better with significant lower overheads than periodic routing protocols in many situations.

In DSR, when a node has a packet to send to some destination and does not currently have a route to that destination in its Route Cache, the node initiates Route Discovery to find a route. The initiator transmits a ROUTE REQUEST packet as a local broadcast, specifying the target. Each node receiving the ROUTE REQUEST, if it has recently seen this request from that particular initiator, discards the REQUEST. Otherwise, it appends its own node address to a list in the REQUEST and rebroadcasts the REQUEST. When the ROUTE REQUEST reaches its target node, the target sends a ROUTE REPLY back to the initiator of the REQUEST, including a copy of the accumulated list of addresses from the REQUEST. When the REPLY reaches the initiator of the REQUEST, it caches the new route in its Route Cache.

Route Maintenance is the mechanism by which a node sending a packet along a specified route to some destination detects if that route has broken, for example because node movement. DSR is based on *source routing*: when sending a packet, the initiator lists in the header of the packet the complete sequence of nodes through which the packet is to be forwarded. Each node along the route forwards the packet to the next hop indicated in the packet's header, and attempts to confirm that the packet was received by that next node. If, after a number of local retransmissions of the packet, a node in the route is unable to make this confirmation, it returns a ROUTE ERROR to the original source of the packet, identifying the link from itself to the next node as broken. The sender then removes this broken link from its Route Cache. For subsequent packets to this destination, the sender may use any other route to that destination in its Cache, or it may attempt a new Route Discovery for that target if necessary.

There are a number of improvements to this basic definition of DSR, but in this project I will try to authenticate nodes and secure the route discovery and maintenance phases.

2.2. SUCV (Statistically Unique Cryptographically Verifiable)

There is a fundamental problem in handling source routing that allows hosts to modify hoe other hosts route packets to a certain destination. The problem is that these operations can be misused by rogue nodes to redirect traffic away from its intended destination. Authentication alone does not solve this problem. Even if a node identifies itself, this has no bearing on its rights to modify how packets to any given address are routed. This is true even if its packets currently seem to originate from the address in question.

Hence protection against hijacking of valid addresses requires cryptographic authorization for operations that modify routing. One way of doing this is by showing that the requesting node owns the address for which routing information is being modified.

In an environment in which the nodes inherently distrust each other, and in which a global or centralized public key infrastructure or key distribution center is not available establishing trust is not an easy task.

In the absence of a third party, proving ownership of identity to another peer is nontrivial. We must note that usual owner verification relies on a third party to provide this function. In **SUCV**, the principal self-generates private/public key pair. However, it is much more practical for protocols to use fixed length identifiers. Because of this, we do not use the public key itself as the identifier. Instead, the principal uses material obtained via a pseudo random function of the public key as its identity (or as part of its address), and proves its ownership by signing it with its private key. The recipient verifies the signature, and, consequently, the ownership of the identity. Hence, redirect operations are only allowed to affect routing for entities which have the SUCV property.

2.3. One-Way Hash Chains

One-way hash chains, or simply one-way chains, are a frequently used cryptographic primitive in the design of secure protocols. We create a one-way chain by selecting the final value at random, and repeatedly apply a one-way hash function H. In my scheme, from the viewpoint of usage, the first value of the chain is the last value generated, and the initially randomly chosen value is the last value of the chain used. One-way chains have two main properties (assuming H is a cryptographically secure one-way hash function):

- Anybody can authenticate that a value v_j really belongs to the one-way chain, by using an earlier value v_i of the chain and checking that $H^{j-i}(v_j)$ equals v_i.
- Given the latest released value v_i of a one-way chain, an adversary cannot find a later value v_j such that $H^{j-i}(v_j)$ equals v_i.

These two properties result in authentication of one-way chain values: if the current value v_i belongs to the one-way chain, and we see another value v_j with the property that $H^{j-i}(v_j)$ equals v_i, then v_j also originates from the same chain and was released by the creator of the chain.

2.4. Diffie-Hellman Key Agreement

The Diffie-Hellman key agreement protocol (also called exponential key agreement) was developed in 1976. The protocol allows two users to exchange a secret key over an insecure medium without any prior secrets.

The protocol has two system parameters p and g. They are both public and may be used by all the users in a system. Parameter p is a prime number and parameter g (usually called a generator) is an integer less than p, which is capable of generating every element from 1 to p-1 when multiplied by itself a certain number of times, modulo the prime p.

Suppose that Alice and Bob want to agree on a shared secret key using the Diffie-Hellman key agreement protocol. They proceed as follows: First, Alice generates a random private value '*a*' and Bob generates a random private value *b*. Then they derive their public values using parameters p and g and their private values. Alice's public value is $g^a \pmod p$ and Bob's public value is $g^b \pmod p$. They then exchange their public

values. Finally, Alice computes $k_{ab}=(g^b)^a$ (mod p), and Bob computes $k_{ba}=(g^a)^b$ (mod p). Since $k_{ab}=k_{ba}=k$, Alice and Bob now have a shared secret key *k*.

The protocol depends on the discrete logarithm problem for its security. It assumes that it is computationally infeasible to calculate the shared secret key $k=g^{ab}$ (mod p) given the two public values g^a (mod p) and g^b (mod p) when the prime p is sufficiently large.

2.5. TESLA

TESLA is a broadcast authentication scheme proposed by Perrig et al. The main technique used in TESLA is a one-way key chain along with delayed key disclosure. After initiating an authentic key from a one-way key chain between the sender and its receivers using a digital signature, TESLA uses MAC (Message Authentication Code) for subsequent broadcast authentications but with delayed key disclosure. In the basic scheme of TESLA, a sender uses a key K from its key chain as the MAC key to compute a MAC over packet P(i), and then attaches the MAC to P(i). The disclosure of the key K is in the *next packet* P(i + 1), which allows the receivers to verify the authenticity of K and hence the MAC of P(i). If both K and the MAC are correct, and if the packet P(i) is guaranteed to be received before P(i + 1) was sent, the receivers then believe P(i) is authentic.

From this description, we can see clearly that the central security issue in TESLA is a receiver's ability to determine the sending time of each packet. This is called security condition in TESLA. TESLA solves this issue through periodical key disclosure and loose time synchronization. In TESLA, time is divided into many equal length intervals. In each time interval the sender discloses one key from its key chain. For example, if the start time is s, the time interval is T, the sender can publish K(i) at time s + i * T. The receivers are required to be loosely synchronized with the sender. Here "loosely" means they know the upper bound of the synchronization error between any two nodes. Given this synchronization error and the sender's MAC key disclosure interval, a receiver can determine whether the sender has already published the MAC key for a packet it just received. If its MAC key is (possibly) already disclosed, the receiver will drop the packet; otherwise, it will use the MAC key in this packet to verify an earlier packet, and then buffer this packet until a later packet containing its MAC key arrives.

The necessity for loose time synchronization may be avoided by using pair-wise shared keys, but at the cost of higher key setup overhead.

One potential shortcoming of the original TESLA protocol is that the receivers cannot authenticate a packet immediately on its arrival. This means that buffering may be a problem especially for mobile equipments with limited resources.

3. Secure Routing in MANET

3.1. Assumptions

First, I assume the wireless network links are bidirectional, that is, if node A can hear node B, node B can also hear node A. This is generally true when the nodes use omni-directional antennas and have the similar power levels. We also know that this

assumption is also necessary for most of the current routing protocols and applications to work in wireless networks. It is very reasonable to think that the network may drop, corrupt, reorder, or duplicate packets in transmission.

Second, I assume that the following condition holds: a packet sent by a node is received by a neighboring node before a third node can replay the packet to it, unless the neighbor under consideration has dropped the packet. For example, if node A sends a packet to its neighbor B, then it is assumed impossible that another node P hears the packet, (potentially) modifies it, and re-sends it to node B such that the replayed packet arrives at node B before the original packet does. We assume that if node B did not drop the original packet, this condition would hold.

Third, if there is a trusted certification authority available in the environment, I assume each node has a public key certificate signed by that trusted certificate authority and also an authentic public key of the Certification Authority. These public keys will be very useful to initiate trust in the ad hoc network. The distribution of certificates and keys can be done in any reliable way. An advantage of using public key certificate is its flexibility and scalability; a node can authenticate other nodes independently, and there is no need for them to have pre-shared keys as in symmetric cryptography.

Fourth, I take into consideration that the mobile nodes are relatively underpowered. Public-key operations such as digital signatures are relatively expensive to compute on platforms such as handheld PDAs. The measures of the speed of RSA signing and verifying on various platforms are noticeable. For example, a 512-bit RSA signature generation takes 2 to 6 seconds, whereas a signature verification takes 100 to 200 milliseconds with the public exponent e = 3.

I also assume loose time synchronization in the ad hoc network when we use TESLA as the broadband authentication scheme, so that we can utilize it. Nodes may compensate the clock drift with periodic re-synchronization. Also each node in the network must be able to estimate the end-to-end transmission time to any other node in the network. Note that these are considered only TESLA is being used.

Now I will consider two situations which may be accepted as two most general cases. The first will be the case that if we have no key distribution centers, no public key infrastructure, namely no central authority. The second case, which is easy to follow from the context, will be having a trusted central for key distribution.

3.2. No Central Authority

In this section, we have the situation that there exists no key distribution centers, no public key infrastructure, namely no central authority. In order to initiate the trust relationship among nodes, the idea of unique identifiers, which are influenced by SUCV addresses, will be used. After the initiation of trust relationship, DSR's regular Route Discovery may be used. Since the nodes will be sure that its correspondent node claims and proves it has the correspondent address, there's no need to complicate route discovery phase. One may think that malicious nodes may reside in the routes and behave accordingly, and then after a certain amount of time, they can drop or mislead packets. But this is the issue of Route Maintenance and it will be discussed in Section 4.

The only way for a node to prove ownership of the address it claims, in the absence of any other centralized or global mechanism, is to prove that it created a sequence of statistically unique series of bits. The resultant bit string will be an identifier that is also routable. In order to generate such addresses, we can use the formula from [1]:

Address = hmac-sha-1-128(sha1 (random_number), sha1 (public_key))

Here is the description of information exchange in order to bootstrap trust between a pair of nodes:

message1: The mobile node sends the message (just to initiate the exchange) to its peer node. This message contains a Nonce, N1. The source address of the packet is the mobile node's current correspondent address. Additionally, there will be a very simple negotiation mechanism that works as follows: Optionally, the mobile node will be able to express which encryption technique to use, like Diffie-Hellman or ESP, etc.

message2: The peer node replies with a message that contains the following: nonce that was sent by mobile node, a Diffie-Hellman value (g^ymodp), and Session Key lifetime for negotiation. The peer node may respond to any optional parameter negotiation included by the mobile node in message1, by choosing the algorithms it wishes to support.

When the mobile node receives message2, it verifies that the nonce N1 is the same as what was sent in message1. Then the mobile node generates a Diffie-Hellman value (g^xmodp) and together with the Diffie-Hellman received from peer node, it creates session keys by using a pseudo random function. After that the mobile node sends message3.

message3: The third message contains the following fields: Public key and random number it (the mobile node) has used to generate its unique Address identifier, and a Diffie-Hellman value. This message must be signed by the mobile node's private key (we remember that the public key is used to generate the unique Address identifier).

When the CN receives the sucvP3, it verifies the signature using the included Public key, and then verifies that this Public key and random number produce the unique Address used as part of the sender's address. Then peer node can conclude that the mobile node owns its address. At this point, the peer node makes a note of this Public key and unique Address. The peer node then also computes the session keys.

Beginning from this point, the mobile node and its peer can use the keys that they shared during the phase. If any of the ends of the connection suspects that their session is compromised, then they have to start all over again. The route requests for discovery of the route will then be protected by these keys, and this trust relationship will enable them to route packets securely. As I mentioned before, route discovery of DSR will be used from this point on.

3.3. With Central Authority

In this section we have the assumption of having central authorities for key distribution among the nodes in the mobile ad hoc network. Here, suppose, the initiating node S performs a Route Discovery for target D, and they share the secret keys K_{SD} and K_{DS} obtained from central authority for message authentication in each direction.

Route Request Authentication: In order to convince the target of prudence of Route Request, S (the initiator) includes a Message Authentication Code encrypted by K_{SD} over a random number or a timestamp. Usage of timestamp makes more sense, because by this way, the target can verify the authenticity and freshness of Route Request easily.

Packet Authentication by Each Node: In order to prevent packet injection and packet forging on each step, the packet should be verified as authenticate. Since all traffic packets will be authenticated, we must use a computationally inexpensive technique that can also provide immediate authentication. Like TESLA, this authentication technique is based upon the use of *one-way key chains*. But, unlike TESLA, in order to reduce overhead, I do not use periodic and delayed key disclosure. Also delayed authentication as in TESLA is not appropriate since a packet would be delayed at each node in the path from the source to the destination. Since each node has to buffer the traffic packets it has received until they are authenticated, delayed authentication will lead to large storage requirements at every node.

So, each node that is source of packets generates a one-way key chain that is used for traffic authentication by its immediate neighbors. Every neighbor of a node will obtain an authentic key in this chain when it first establishes a trust relationship with the node. A node transmitting a packet will append a new key from the chain to the packet. All the neighboring nodes receiving this packet can verify the authenticity of this packet by checking the validity of the attached key.

Also one another important issue is, an attacker can remove a node from the node list in a Request, so only authenticating the data may not be enough. What we can do is, again using one way hash functions; we can verify that no node was omitted. In order to achieve this, I assume that every node has a one way key chain and that all nodes know an authentic key of the one way key chain of each other node.

Route Discovery: As in section 3.2, after establishing trust relationship, discovery of routes will be very similar to that of DSR. The slight differences will occur for the purpose of authentication. Here is a brief description how route discovery will work:

A Route Request packet will contain: Route Request, initiator, target, id, hash chain, node list, and MAC list. The initiator and target are set to the address of the initiator and target nodes, respectively. As in DSR, the initiator sets the id to an identifier that it has not recently used in initiating a Route Discovery. The initiator of the Request then initializes the hash chain to $MAC_{K_{SD}}$ (initiator, target, and id) and the node list and MAC list to empty lists.

When any node A receives a Route Request for which it is not the target, it checks for initiator and id for a recent occurrence. If so, it discards the packet. Otherwise, the node modifies the Request by appending its own address, A, to the node list in the Request, replacing the hash chain field with H [A, hash chain], and appending a MAC of the entire Request to the MAC list. The node uses a key from one way key chain that is generated

for traffic authentication compute the MAC. Finally, the node rebroadcasts the modified Request, as in DSR.

When the target node receives the Route Request, it checks the validity of the Request by determining the traffic keys of immediate neighbors and hash chain filed should be equal to

$$H\,[Node_n, H\,[Node_{n-1}, H\,[\ldots.., H\,[Node_1, MAC_{K_{SD}}(initiator, target, id)]\ldots.]]]$$

After determining that this Request is valid, it returns a Route Reply message containing Route Reply, target, initiator, node list, MAC list, target MAC. The target MAC set to a MAC computed with K_{DS}.

This message is returned to initiator of the Request along the source route obtained by reversing the sequence of hops in the node list of the Request. When the initiator receives a Route Reply, it verifies that the target MAC is valid, and that each MAC in the MAC list is valid. If these tests succeed, the node accepts the Route Reply; otherwise, it discards it.

4. Route Maintenance

In the absence of a *central authority*, route maintenance will be achieved as it is done in DSR. If a node detects that a route has broken, because of node movement or some other reason, the sender removes the broken link from its Route Cache. The source lists in the header of the packet the complete sequence of nodes through which the packet is to be forwarded. Each node along the route forwards the packet to the next hop indicated in the packet's header, and attempts to confirm that the packet was received by that next node. If, receipt of this confirmation fails, it returns a Route Error to the original source of the packet, identifying the link from itself to the next node as broken. The sender then removes this broken link from its Route Cache. For subsequent packets to this destination, the sender may use any other route to that destination in its Cache, or it may attempt a new Route Discovery for that target if necessary.

In the existence of a *central authority*, the route maintenance issue is easier to handle. Again it will be based on DSR's route maintenance. A node forwarding a packet will return a Route Error to original sender after a certain number of attempts to transmit the packet to the next hop along the route. In order to prevent unauthorized nodes from sending Errors, the Error messages will be authenticated just like traffic packets. With periodic messages, nodes can broadcast to their immediate neighbors their new key from the one way key chain. By this way, two things will be achieved: first one is, nodes will understand which of their neighbors are alive and the second is capturing and using a traffic key from that key chain will become harder because of these updates.

Since we are using DSR, it is very likely to have multiple routes from a source node to destination. In order to prevent some kind of attacks and increase the throughput, it will be better to use more convenient routes. We may choose routes based on their performance in packet delivery. Introducing mechanisms that penalize specific nodes for

routing misbehavior is subject to blackmail attack, where a sufficient number of attackers maybe able to victimize a decent node.

In order to achieve this, a node with multiple routes to a single destination can assign a fraction of packets that it originates to be sent along each route. When a smaller fraction of packets sent along any particular node are successfully delivered, the node can begin sending a bigger portion of split packets to that destination along that route. So, this may be understood as ranking the routes, and among those routes, the most efficient or the one with the highest throughput is selected. Hence, if a malicious node drops packets in order to harm ongoing communication, the scheme will adept itself accordingly.

5. Dynamically Joining and Leaving Nodes

Again we will consider the situation of dynamically joining and leaving nodes from two points of view. The first is the absence of a central authority. We must note that in the absence of a central authority, all route discoveries and maintenance depends on initial trust bootstrap. After this, the regular algorithms for DSR with slight modifications are being used. These modifications are of course for the authenticity and confidentiality of the communication.

So, if a node wants to join ongoing communication, the only issue it has to meet is providing a valid and a unique identifier (address notion in 3.2). Then the trust initialization with the immediate neighbors will take place. At the end of this stage, the new node will lie on routes from various sources to destinations. When the node leaves a neighborhood because of movement, low battery power or any other problem, the route maintenance part will take care of it. After a certain amount of time, that node will be removed from route caches.

The second situation is, when we have a central authority. In this case, the central authority delivers keys to nodes that wish to communicate. And these keys are the only source of trust initialization. So, if a node wants to join a network and communicate other nodes, it should first acquire keys from system's central authority. Of course this is only possible if the new node is being recognized by central authority. After this step, since the new node has keys, it can participate in route discovery and packet forwarding phases of communication. As we remember, the nodes periodically release keys from their one way key chain, in order to acknowledge their immediate neighbors that they are still there and they are who they claim to be. Hence, if a node does not receive a valid key like this from a neighbor within a certain amount of time, it will conclude that there is a problem with that node (dead because of low battery or out of range because of movement). Then it will terminate immediately its trust with this neighbor.

6. Attacks and Defenses

6.1. Forging / Dropping Packets

If a malicious node behaves decent in route discovery and in message transmission phase starts dropping the packets, in both situations (absence and existence of central authority) we will not be affected seriously. That is because of our route maintenance scheme. Only a few packets may be dropped, but then we will detect that compromised route and stop using it.

There is another problem such that a malicious node can drop the packets and then try to acknowledge the source as if the message is received by the target. In the situation of having a central authority, source will be able to detect the received packet's origin. So it will not be affected. But, in the other case, where we don't have central authority, we can only detect the authenticity of messages from immediate neighbors. The malicious node can deceive one of its immediate neighbors, and the false acknowledgement will be propagated up to us from one of our immediate neighbors. So this scheme is vulnerable if a malicious node drops the packets and sends fake acknowledgements.

6.2. Man-in-the-middle

In the case of having no central authority, the nodes should establish a trust relationship first. At the end of initialization of trust, each end will have the shared key, and from then on they will use that key in between their message exchanges.

Hence this scheme is vulnerable to man-in-the-middle attacks during the bootstrap of trust phase. If a malicious node can hear both mobile and its peer node, it can detect the key they share to communicate and eavesdrop the subsequent messages.

But in the case of having a central authority, the nodes will have the keys in the beginning of communication and all subsequent route and traffic packet will be encrypted. Hence it will disallow the malicious nodes to eavesdrop the messages.

6.3. DoS

When we don't have a central authority, malicious nodes can try to attack a host by sending storm of message1, or message2 or message3. In the case of a storm of message1 the host does not change its state, and replies with a constant message containing a nonce. The host only changes the state, if it receives a valid response to its challenge. In order to prevent message2 storms, again nonce can be used. If the host receives a message2 with a nonce that is not equal to the nonce it has sent, this message2 is rejected.

If have a central authority, the issue is different. Our network may be flooded with many Route Requests. Since the source address of each Request is authenticated, and since each new Route Discovery needs to carry a new one way Route Discovery chain value, the malicious node can only produce Route Requests with its own source address. An upper bound on the sending rate can be enforced by limiting of Requests at each node.

6.4. Wormhole

A wormhole attack is launched by multiple colluding attackers. The attackers X and Y have a private channel that allows them to communicate directly. X forwards every message it eavesdropped from source node S, including one way key chain update messages and traffic packets, to Y through the wormhole. X then rebroadcasts the update messages of S's new key and modify the traffic packets to deceive target node D. Because D may receive the original key update from S later then this fake broadcast, it may still accept the replayed key, and will forward the modified traffic packets for node Y.

Packet leashes alone cannot prevent this attack, but there may be a possible solution such that each node forwarding a Route Request includes its GPS coordinates in that Request, then a node can detect if it should be reachable from the previous hop, and if the hop before the previous hop should be able to reach the previous hop. If both of these checks succeed, then the scheme is safe against such attacks. But, I must admit that having GPS equipped by nodes is a strong assumption.

7. Conclusion

In this project I tried to provide a secure routing scheme for mobile ad hoc networks by combining the situations for availability and absence of a central authority. The assumptions that I made are reasonable for their cases. If somehow we can make these two scenarios work together, then we will have an adaptive secure routing scheme. It will work for both, existing and non-existing central authorities.

Basically, in the absence of a central authority, a node should prove who it claims to be. In order to do this, it creates a sequence of statistically unique series of bits. The resultant bit string is an identifier that is also routable. After that, it initializes trust relationships with neighbors enabling them to work together in Route Discovery phase. The discovery and maintenance of routes are inherited from DSR. In the context of security analysis this scheme may be called fairly safe. Bootstrap of trust phase is its vulnerable part.

Then when we look at the case that we have a central authority, each node has keys to authenticate one another's route discovery and traffic packets. The route discovery and maintenance are very similar to DSR's route discovery and maintenance processes with slight differences. MACs and keys from one way hash key chains are added in order to provide authenticity. From the security analysis point of view, this scheme is safe against forged/dropped packets, man-in-the-middle and DoS attacks, and also with a strong assumption of being equipped with GPS, it also prevents wormhole attacks.

8. References

[1] Gabriel Montenegro and Claude Castelluccia. Statistically unique and cryptographically verifiable identifiers and addresses. *In Proc. 1SOC Symposium on Network and Distributed System Security (NDSS 2002), San Diego, February 2002*

[2] Vikram Gupta, Srikanth Krishnamurthy, Michalis Faloutsos, Denial of Service Attacks at the MAC Layer in Wireless Ad Hoc Networks *IEEE Milcom 2002, Anaheim, California, October 7-10, 2002*

[3] Srdjan Capkun, Jean-Pierre Hubaux, Levente Buttyan Mobility Helps Security in Ad Hoc Networks. *The Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing Annapolis, Maryland, USA June 1-3, 2003*

[4] Panagiotis Papadimitratos and Zygmunt Haas Secure Data Transmission in Mobile Ad Hoc Networks *ACM Workshop on Wireless Security (WiSe 2003) September 19, 2003 Westin Horton Plaza Hotel, San Diego, California, U.S.A*

[5] Yih-Chun Hu, Adrian Perrig, David B. Johnson. A secure On-Demand Routing Protocol for Ad hoc Networks *MobiCom 2002, September 23-28, 2002, Atlanta, Georgia, USA*

[6] P. Papadimitratos and Z.J. Haas, Secure Link State Routing for Mobile Ad Hoc Networks, *IEEE Workshop on Security and Assurance in Ad hoc Networks, in conjunction with the 2003 International Symposium on Applications and the Internet, Orlando, FL, January 28, 2003*

[7] S. Zhu, S. Xu, S. Setia and S. Jajodia. LHAP: A Lightweight Hop-by-Hop Authentication Protocol For Ad-Hoc Networks, *ICDCS 2003 International Workshop on Mobile and Wireless Network (MWN 2003), May 2003*

[8] Lakshmi Venkatraman and Dharma P. Agrawal A Novel Authentication scheme for Ad hoc Networks *Wireless Communications and Networking Confernce (WCNC 2000), IEEE, Pages 1268-1273, vol.3*

[9] Levente Buttyfin and Jean-Pierre Hubaux. Report on a Working Session on Security in Wireless Ad Hoc Networks. *Mobile Computing and Communications Review, Volume 6, Number 4. November 2002.*