A Distributed Security Architecture for Ad hoc Networks

Ratan Guha[1], Mainak Chatterjee[2] and Jaideep Sarkar[2]

[1] School of Computer Science
University of Central Florida
Orlando, FL 32816-2362

[2] Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2450

Abstract. Secure communication in ad hoc networks is an inherent problem because of the distributiveness of the nodes and the reliance on cooperation between the nodes. All the nodes in such networks rely and trust other nodes for forwarding packets because of their limitation in the range of transmission. Due to the absence of any central administrative node, verification of authenticity of nodes is very difficult. In this paper, we propose a clusterhead-based distributed security mechanism for securing the routes and communication in ad hoc networks. The clusterheads act as certificate agencies and distribute certificates to the communicating nodes, thereby making the communication secure. The clusterheads execute administrative functions and hold shares of network keys that are used for communication by the nodes in respective clusters. Due to the process of authentication, there are signalling and message overheads. Through simulation studies, we show how the presence of clusterheads can substantially reduce these overheads and still maintain secure communication.

1 Introduction

The absence of centralized administration or fixed network infrastructure is the essence of peer to peer or ad hoc networks. Such networks can be established where there is no infrastructure or in which the existing infrastructure does not meet requirements such as deployment delay and costs. Though ad hoc networks provide huge benefit to applications like military operations, they at the same time fail to provide reliable and secure communications. Most of the research in ad hoc networking have assumed non-adversarial network setting or a trusted environment. Relatively little research has been done in a more realistic setting in which an adversary may attempt to disrupt the communication. A central issue concerning the design of any service in ad hoc networks is not to rely on any centralized entities, because such entities would obviously be prone to attacks, and also their reachability could not be guaranteed at all times for all participants of the network. Therefore, it is not possible to implement a centralized, trusted entity for managing network keys of the participants as performed in conventional wired networks, where a central trusted authority is always capable of managing the network and providing the entities of the network with keys for authentication purposes.

In this paper we propose a solution where we make the clusterheads as the distributed *certificate authorities* (CA) who will be vested with the charge of storing network keys and then distributing those keys to the communicating nodes after their authenticity has been confirmed by the clusterheads.

The rest of the paper is organized as follows. In section 2, we discuss the general problem of securing ad hoc networks, with common attack techniques and methods to tackle them. In section 3, we propose our clusterhead based architecture and present the mechanism through which both clusterheads and ordinary nodes are authenticated. We present the simulation model in section 4 and show how the overheads vary with the number of hops. Conclusions are drawn in the last section.

2 Securing Ad hoc Networks

Unlike wired networks where an adversary must gain physical access to the network wires or pass through several lines of defense at firewalls and gateways, attacks on the wireless network can come from all directions and target at any node. The use of wireless links renders the network vulnerable to attacks ranging from passive eavesdropping to active interfering. Moreover, mobile nodes with inadequate physical protection are easy to be captured, compromised and hijacked. Thus attacks by a compromised node from within the network are much damaging and hard to detect. In ad hoc networks, the network algorithms rely on the cooperative participation of all nodes and the infrastructure. The lack of centralized authority means that the adversaries can exploit this vulnerability for new types of attacks designed to break the cooperative algorithms. The mobility of nodes on the other hand requires sophisticated routing protocols. Thus security is an additional problem as it is hard to identify incorrect routing information generated by compromised nodes or as a result of some topology changes. With slow link, limited bandwidth, battery power constraints, nodes are inclined to adopt mechanisms like disconnected operation and location-dependent operations. In a nutshell, wireless ad hoc networks will not have a clear line of defense and every node must be prepared to encounter any adversary directly or indirectly.

2.1 Types of Attacks on Ad hoc Networks

Attacks on ad hoc networks usually try to disrupt the routing protocols. Such attacks fall in two categories: *routing disruption* and *resource consumption* [3]. In a routing disruption attack, the attacker attempts to cause legitimate data packets to be routed in a dysfunctional way. In a resource consumption attack, the attacker injects packets into the network and attempts to consume valuable network resources as memory storage, battery life and computation power. Both attacks are instances of denial-of-service attack. An example of routing disruption attack will be when an attacker sends forged routing packets to create a routing loop, causing packets to traverse nodes in a cycle without reaching their destinations, consuming energy and the available bandwidth thereby clogging the network. An attacker may also create a *black hole*, in which all packets will be dropped. Therefore by creating a false route into the black hole all the packets will be discarded by the attacker thereby depleting the network resources. The attacker may also attempt to use *detours* or *partition* the network thereby preventing a particular set of nodes to be part of a route. The attacker may also include itself into the route and along with that add some other nodes in the route that otherwise would not have been necessary and thereby using the resources more than required. In case of resource consumption attacks, the attacker tries to deplete the resources of the network. It tries to inject extra packets into the network, which may consume even more bandwidth or computational resources as other nodes process and forward such packets.

2.2 General Techniques for Authentication and Secret Sharing

In securing a network, goals like authenticity, integrity, confidentiality, non-repudiation and availability are most important. Authentication of communicating entities is of particular importance as it forms the basis for achieving the other security goals: e.g., encryption is not worthwhile if the communicating partners have not verified their identities before. The reason being that if the communicating nodes are not authenticated then a malicious node might join the network and participate in communication thereby making the use of encryption futile. Authentication of entities and messages can be realized in different ways using either symmetric (3DES, AES) or asymmetric (ElGamal, RSA) [11] cryptographic algorithms. Symmetric algorithms depend on the existence of a pre-shared key (which does not exist in the general case). Authentication by asymmetric cryptography requires a secure mapping of public keys to the owner's identities which is often accomplished by by public key infrastructures (PKI). PKI's use digitally signed certificates to verify a key owner's identity. Each user has to prove his identity to a certification authority (CA) and after the authority has authenticated the user it provides him with a public key with which it can then communicate with other users who in turn will have to go through the same process.

In contrast to fixed networks, a centralized PKI or even a centralized certification authority is not feasible in ad hoc networks, due to the lack of infrastructure in these kind of networks. Distributing the signing key and the functionality of a CA over a number of different nodes by the means of secret sharing and threshold cryptography is a possible solution to this problem, Secret sharing schemes realize confidentiality of a cryptographic secret by spreading it across different entities. As secret sharing schemes need no central authorities, they are predestined for ad hoc networks. One secret sharing scheme is threshold cryptography: A trusted dealer divides a secret into n parts so that the knowledge of k parts $(k \leq n)$ allows the reconstruction of the secret, which is not possible with the knowledge of $k - 1$ or fewer parts. This is called a (k, n) threshold scheme [10]. In general, a trusted dealer is a central authority and thus another central target for attacks. To avoid this, the participants have to construct the secret without any central authority. The construction algorithm has to ensure that participants can only transmit correct values and that each participant can verify both secret and shares, which is called verifiable secret sharing [9]. Due to the movement of mobile nodes, the topology of ad hoc networks changes frequently, and moreover, nodes can join or leave the network at any time. Hence, an algorithm for distributing the same key to a different set of participants is required. Such a refresh algorithm [5] can be triggered periodically, event-based, or both. Another form of symmetric cryptographic algorithm is TESLA [3] where the authors have authenticated the nodes using time. When using this protocol each node will be knowing the time it takes to send a message to any other intermediate node in the route. A time-stamped value will then be computed using a Hash Function known to all the nodes and then after each passing interval these values will be made public to all the nodes.

3 Proposed Clusterhead based Authentication Architecture

Clusterheads are certain nodes in the network that are selected to do some additional jobs. They are responsible for the formation of *clusters* each consisting of a number of nodes and maintenance of the topology of the network. A clusterhead does the resource allocation to all the nodes belonging to its cluster. Due to the dynamic nature of the mobile nodes, their association and dissociation to and from clusters perturb the *stability* of the network and thus reconfiguration of clusterheads is unavoidable. This is an important issue since frequent clusterhead changes adversely affect the performance of other protocols such as scheduling, routing and resource allocation that rely on it. Choosing clusterheads optimally is an NP-hard problem [1]. However, there are a number of heuristics for choosing clusterheads depending on the objective that the network wants to achieve.

Previously work has been done in this field where clusterheads have been used as certifying agencies [2]. In this paper, we do not try to propose a new clustering algorithm. Instead, we assume that there is a clustering algorithm present which judiciously selects some of the nodes as clusterheads. It can be noted that the clusterheads will solely be used for certification and not for any other purpose like routing or resource allocation. Even after the clusterheads are identified, the topology remains *flat* as opposed to hierarchical nature in usual clusterhead based networks.

3.1 Network Architecture

We consider an ad hoc network consisting of nodes with equal transmission range. When the network is initialized only trusted nodes are present. Of course, there would be malicious nodes during the lifetime of the network. Since, our main focus is secure communication, we do not investigate routing protocols and assume that some routing algorithm is present which provides a secure route. Though we use the concept of clusterheads, they do not help in routing as such and there is also no provision for gateways to allow inter-cluster communication. Two nodes although being in different clusters can still communicate with each other if they are within the transmission range. However there has to be a mutual agreement between the two nodes to exchange certificates through the clusterheads which maintain the public keys of all the nodes in its cluster. Every clusterhead will also have information of secret keys with which it can communicate with all other neighboring clusterheads.

Caching: In a session between a source-destination pair, if it so happens that the pair belongs to different clusters then after the key exchange is done between the clusterheads, it will store the information about the public keys of the node that lies in the other cluster in its cache. This caching feature will help the clusterheads in reducing the message overhead if in future there exists a route that involves the node from the other cluster. However this information will not be kept in the cache forever, but will be cleared automatically after a given time period.

Clustering: This process involves partitioning the network to identify certain nodes which will become clusterheads and act as the certificate authorities. Each node has the information about its neighboring nodes with the help of beacon signals. We chose the simplest of clustering algorithms to partition the network into clusters since our prime concern is not clustering efficiently but demonstrate the use of clusterheads as administrators that can help in securing the network efficiently. We assume the highest degree [8] clustering algorithm in which the node with the highest degree is chosen to be the clusterhead. Once a clusterhead is found, all its neighbors join the cluster. The clustering algorithm is invoked till all the nodes have a clusterhead to attach to or itself is a clusterhead. This of course does not guarantee a connected network. Figure 2 shows a few nodes that are isolated.

Inheriting Authority: The clusterheads having been elected are then given the information with which they can create certificates and are also given the information of the public keys of all the other nodes in the network and the private key of the certificate authority itself. Moreover the clusterheads are given pairs of shared secret keys with which they can communicate with neighboring clusterheads securely. Each clusterhead will then assign a pair of public and private keys to all other nodes of the network at the time of formation of the network. The nodes also have the capability to generate new public-private key pairs upon approval from the certificate authority.

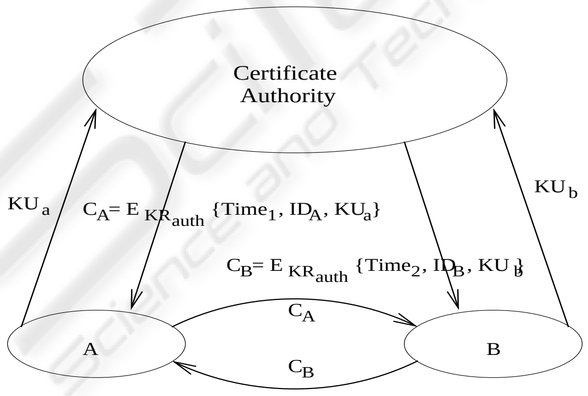
Certificate Authority
KU_a
$C_A = E_{KR_{auth}} \{Time_1, ID_A, KU_a\}$
KU_b
$C_B = E_{KR_{auth}} \{Time_2, ID_B, KU_b\}$
C_A
A
B
C_B

Fig. 1. Protocol followed in key exchange

3.2 Authentication Mechanism

Idea to use a distributed certification authority based on a shared certification key and threshold cryptography for securing ad hoc networks was first proposed by Zhou and Haas [12]. It was further developed in the COCA system [13], where a general distributed authentication service was proposed. Recently similar work was been done by Bechler *et. al.*[2]. Our approach is based on the same underlying principles, but introduces several new concepts like a cluster-based network structure where the clusterheads are only responsible for the authentication mechanism. Clusterheads would authenticate new nodes entering the network and are also able to discharge their responsibilities to newly formed clusterheads. However such a situation will arise only when the network topology changes due to the mobile nature of the nodes. Let us now discuss the securing scheme.

The approach of certificate authorities by Kohnfelder [11] suggested the use of certificates that can be used by nodes to exchange keys without contacting a public-key authority. Each certificate containing a public key and other information is created by the certificate authority and is given to the node with matching private keys. A node will convey its key information to the other nodes by transmitting the certificate. The other nodes can then verify that the certificate was actually created by the certificate authority and not by any other malicious node. There are few requirements for this scheme to work, which are given below.

• Any node can read a certificate to determine the identity and public key of the certificate's owner.
• Any node can verify whether the certificate originated from the certificate authority.
• Only the certificate authority can update the certificates.
• The certificate authority can also transfer its duties to any other node once it has authenticated that the node is a *true* node.
• Any node can also verify the currency of the certificate implying if the certificate is of the latest version or not.

The general scheme [11] by which a certificate authority (CA) secures a system is shown in figure 1. For node A, the certificate authority provides a certificate of the form

$$C_A = E_{KR_{auth}}[T, ID_A, KU_A]$$

where $E_{KR_{auth}}$ is the encryption algorithm of the certificate authority using its private key, once node A has made a request for the certificate. T is the timestamp that validates the currency of the certificate. ID_A and KU_A are the *node-ID* and the public key of node A respectively. Once node A receives the certificate C_A, it passes the certificate to the node with which it wants to communicate (say B). On receiving C_A, B will read and verify the certificate as follows.

$$\begin{aligned} D_{KU_{auth}}[C_A] &= D_{KU_{auth}}[E_{KR_{auth}}[T, ID_A, KU_A]] \\ &= (T, ID_A, KU_A) \end{aligned}$$

The recipient node B will use the public key of the certificate authority KU_{auth} with the decryption algorithm D to decrypt the certificate. Because the certificate is readable only using the certificate authority's public key, it confirms that the certificate came from the certificate authority and not any other malicious node. The parameters ID_A and KU_A provide the node B with the name and public key of the certificate's holder, which is node A.

The timestamp T validates the currency of the certificate and also secures a communication even if a node A's private key is known by an opponent. A generates a new private-public key pair and applies to the certificate authority for a new certificate. Meanwhile, the opponent will replay the old certificate to B. If B then encrypts messages using the compromised old public key, then the opponent will read those messages. With the use of the timestamp this kind of a situation will be taken care of as the timestamp will be indicating the currency of the certificates. Thus we see that finally at the end of the transmission the node B knows the public key of A without the other nodes in the network knowing. In a similar fashion node B will also communicate with the certificate authority and the same sequence will follow by which node A will know the public key of node B. Once this process is complete, both nodes can communicate either way by encrypting the messages with their public keys and decrypting with the private keys.

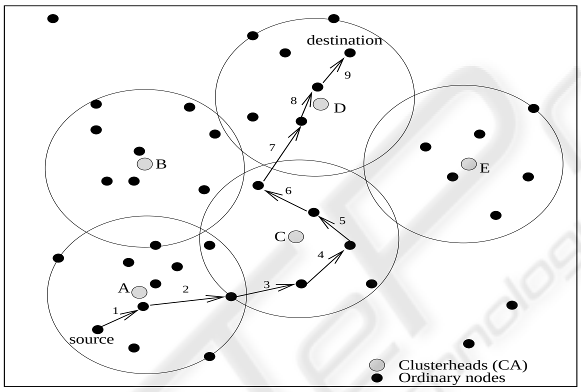

Fig. 2. Proposed clusterhead based architecture

3.3 Securing the Message Communication

In our architecture once the network has been formed and a route has been discovered the authentication mechanism is initiated. From the route information, the clusterhead identity for all the nodes participating in the route is obtained. Our main aim is to secure the route on a hop by hop basis. The source first communicates with its next hop neighbor in the route and then goes through the entire security protocol as explained in section 3.2. For example, the source will first communicate with its clusterhead and then request for a certificate for communication as

$$Request_{source} = E_{KR_{source}} E_{KU_{auth}}[node - ID].$$

Within this request, the node will also send the *node-ID* of the node with which it wants to communicate. The source encrypts this message first with the public key of the certificate authority and then with its private key. The certificate authority will decrypt the message first with the public key of the source and then with its private key as

$$D[Request_{source}] = D_{KU_{source}} D_{KR_{auth}}[node - ID]$$
$$= node - ID$$

In this way the certificate authority can verify that the message actually came from the source and not from any malicious node. Once the certificate authority knows the *node-ID* of the node with which the source wants to communicate it scans through its cluster to see if that node belongs to that cluster. Two scenarios might arise; the node might be in the same cluster or in a different one. If it is in the same cluster the problem becomes simple. However, if the node is not in the same cluster the problem becomes complex as the certificate authority (say $CA1$) will have to find and exchange information with the new clusterhead (say $CA2$) to which the other node belongs.

Let us now explain the communication process. As shown in figure 2 we observe that there are both intra-cluster and inter-cluster routes. The *source* belongs to clusterhead A whereas the *destination* is in cluster D. The route segments 3 and 7 demonstrate inter-cluster routing. In case of segment 3 there will be a mutual handshaking between clusterheads A and C, whereas for segment 7 there is a handshake between clusterheads C and D. This procedure of handshaking between two different clusterheads starts with the clusterhead asking the source about the affiliation of the node in the next hop. Once the clusterhead $CA1$ gets that information it will send a query with a timestamp encrypted with its secret key to the new clusterhead via the source and its next hop neighbor. The message is as follows.

$$M_{CA1} = K_{secret}[N1, node - ID, T1]$$

The message will essentially ask if the next hop neighbor actually belongs to that cluster. The message consists of the node-ID of the next hop neighbor, a timestamp $T1$, and a nonce $N1$. The new clusterhead ($CA2$) after decrypting the message with the shared secret key will authenticate that the message essentially came from another clusterhead and not from any other malicious node. Once the query is verified, the clusterhead searches its own cluster for the target node. If found, the clusterhead sends the public key of that node. The message is encrypted with the shared secret key between the two clusterheads and the message is as follows.

$$M_{CA2} = K_{secret}[N1, N2, T2, KU_{node-ID}]$$

where, $N1$ is the nonce generated by $CA1$, $N2$ is the nonce generated by $CA2$. $T2$ is another timestamp and $KU_{node-ID}$ is the public key of the target node. Once $CA1$ receives this message it will re-confirm to $CA2$ by sending another message including the nonce generated by $CA2$ and will also send the public key of the source to $CA2$. The message that $CA1$ sends will be as follows.

$$M_{CA1} = K_{secret}[N2, T3, KU_{source}]$$

Thus $CA2$ knows the public key of the source.

Our architecture also proposes that once a clusterhead knows a public key of any other node that does not belong to its cluster, then that key is stored for use in future sessions. In this way after a period of time, the stored information about the public key will bring down the message overhead. Once this handshaking procedure is done the respective clusterheads will provide an encrypted certificate having the public keys of the other node with which the respective node of the cluster wants to communicate as explained in the protocol in section 3.2.

The clusterheads will essentially send a certificate to the node in its cluster, for example in our case clusterhead A will send a certificate to the source as shown in figure 2 as below

$$C_{source} = E_{KU_{source}}[T1, KU_{next-hop-neighbour}]$$

Once the source receives the public key of the next-hop-neighbor with which it has to communicate, it first decrypts the certificate with its private key and obtains the public key of the next-hop-neighbor as shown below.

$$M = D_{KR_{source}}[C_{source}] = [T1, KU_{next-hop-neighbour}]$$

The next-hop-neighbor on the other hand will go through the same procedure and receive a certificate bearing information about the public key of the source from its clusterhead.

Once the nodes receive each other's public keys they start the message communication encrypted with the public key of the other node to which the message is sent and the receiving node decrypts it with its private key as that is known to the nodes at the time of setup of the network. This procedure propagates throughout the route until the destination is reached.

4 Simulation Model and Results

We simulated the proposed clusterhead based authentication mechanism on a UNIX based environment. Nodes were randomly scattered on a square grid of 100 $\times$ 100. Clustering was performed and some routes were obtained based on some random source-destination pairs. We implemented the DSR[6] routing algorithm to find a source destination pair. Once the route between the source-destination pair is obtained the next task is to ensure a secure message transmission between the source and destination. For routes of various length, we calculated the message overhead. For the same route length, the actual message overhead will depend on the distribution of the nodes along the route, i.e., how many nodes belong to the same cluster and how many do not. Moreover there is a dependence of message overhead on time as the caching feature of the clusterheads come into play as with time, the key informations of more and more neighboring nodes will be cached.

4.1 Results

We base our results on the message overhead against the number of hops in a route and show the variation in figure 3 and figure 4. The message overhead is the additional bytes of information sent by either the clusterhead to nodes or vice versa in exchange of certificates as well as the exchange of the encrypted data between the nodes. This

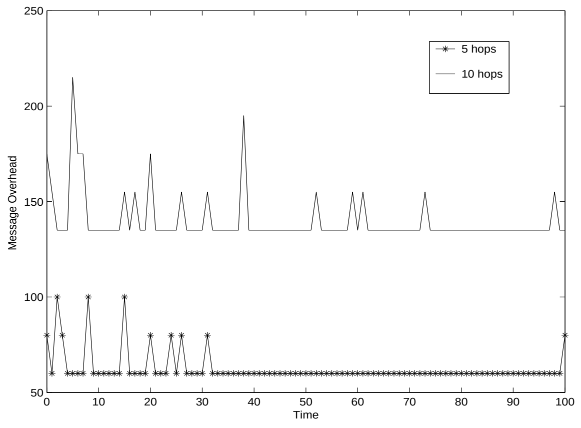

Fig. 3. Message overhead

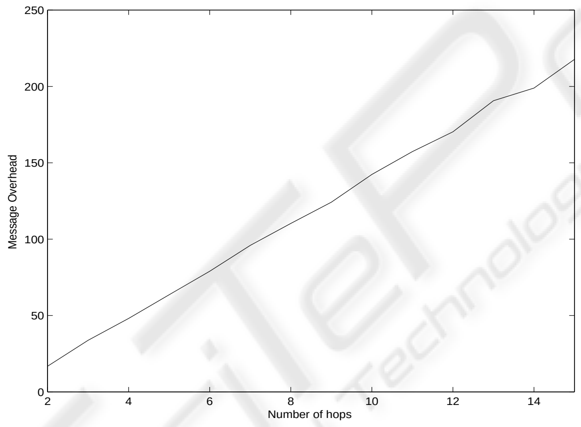

Fig. 4. Message overhead

overhead will be more in case of inter-cluster message transmission as this would involve clusterheads communicating amongst themselves for mutual authentication and also for transferring the public keys. Figure 3 gives the overhead against a hop length of 5 and 10 as time evolves. For both cases, there is a minimum overhead. However, at times, the overhead spikes. This is because of mobility and re-authentication of nodes in the route. Also, the fluctuation is due to the distribution of the nodes- within same clusters or different ones.

We also observe that as time evolves there are few aberrations of the message overhead and it tends to be at the minimum. This is because of the cache of the clusterheads which slowly gathers information about the public keys of various nodes and become richer in information content. The gathered information avoids extra overhead in inter-cluster communication. It is obvious that the minimum message overhead for routes of length 10 would be more than that of routes of length 5.

In figure 4, we observe that when the number of hops increases the message overhead is almost linear. The reason being that with time the overhead against the number of hops averages out as the caching reduces the overhead. So even if there are sudden increases in the overhead, the average is considerably lower.

5 Conclusion

With ad hoc networks becoming increasing popular, securing communication in such networks is gaining importance. Securing ad hoc networks is more challenging because of the absence of an central authority and the distributiveness of the nodes. In this paper, we propose a clusterhead based distributed authentication mechanism. The clusterheads execute administrative functions and act as certificate agencies and distribute certificates to the communicating nodes, thereby making the communication secure. Simulation experiments were conducted to study the message overhead that would be incurred due to the process of authentication. The use of cache to store public keys has shown to better the performance in the long run.

References

1. S. Basagni, I. Chlamtac, and A. Farago, “A Generalized Clustering Algorithm for Peer-to-Peer Networks”, Proceedings of Workshop on Algorithmic Aspects of Communication (satellite workshop of ICALP), July 1997.
2. M. Bechler, H. J Hof, D. Kraft, F. Pahlke, L. Wolf “A Cluster Based Security Architecture for Ad Hoc Networks,” *IEEE Infocom 2004*.
3. Yih-Chun Hu, Adrian Perrig and David B. Johnson, “ARIADNE: A Secure On-Demand Routing Protocol for Ad Hoc Networks” *Mobicom*, Sept. 2002 ACM.
4. Yih-Chun Hu, Adrian Perrig and David B. Johnson, “Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks.” Technical Report, Department of Computer Science, Rice University, December 2001.
5. Y. Desmedt and S. Jajodia, “Redistributing secret shares to new structures and its applications,” George Mason Univ., Tech Report., 1997.
6. D.B Johnson, D.A Maltz and Y.C Hu, “The Dynamic Source Routing in ad hoc wireless Networks,” Imielinski and Korth, editors, *Mobile Computing* volume 353, Kluwer Academic Publishers, 1996.
7. J. Kong, P. Zerfos, H. Luo, S. Lu adn L.Zhang, “Providing robust and ubiquitious security support for mobile ad-hoc networks,” in Proc. 9th International Conference on Network Protocols (ICNP). Riverside, California: IEEE, Nov. 2001, pp 251-261.
8. F.G Nocetti, J.S Gonzalez, I. Stojmenovic, “Connectivity based k-hop clustering in wireless networks,” Telecommunications Systems 18 (2001) 1-3, 155-168.
9. T. Pedersen, “A threshold cryptosystem without a trusted party,” in Advances in Cryptology, Proc. Eurocrypt’91, ser. LNCS, vol 547. Springer-Verlag 1991.
10. A. Shamir, “How to share a secret,” ACM Comm., Vol 22, no. 11, 1979.
11. W. Stallings, “Cryptography and Network Security : Principles and Practices”, 3rd edition, Prentice Hall
12. L. Zhou and Z. J. Haas, “Securing ad hoc networks,” IEEE Network, vol. 13, no. 6, pp 24-30, 1999.
13. L. Zhou, F. B Schneider, and R. van Renesse, “COCA: A secure distributed on-line certification authority,” ACM Trans. Computer Systems, vol. 20, no. 4, pp. 329-368, Nov. 2002.