

Self-Organized Public Key Cryptography in Mobile Ad Hoc Networks

K. Sahadevaiah¹, O.B.V. Ramanaiah²

¹Assistant Professor,
Department of Computer Science & Engineering,
University College of Engineering,
Jawaharlal Nehru Technological University: KAKINADA,
Kakinada – 533 003, Andhra Pradesh, INDIA.
Email: ksd1868@gmail.com

²Professor & Head,
Department of Computer Science & Engineering,
University College of Engineering,
Jawaharlal Nehru Technological University: HYDERABAD,
Jagityala – 505 327, Andhra Pradesh, INDIA.
Email: obvramanaiah@gmail.com

ABSTRACT

A Mobile Ad hoc NETWORK (MANET) is a self-organizing, short-lived infrastructure-free, multi-hop wireless network that contains collection of cooperative autonomous freely roaming mobile nodes communicating with each other by wireless radio links with no human intervention. Each mobile node functions as a specialized router to relay information to other mobile nodes. Self-organization mechanisms create many new and exciting application areas for ad hoc communications. Mobility has a large impact on the behavior of ad hoc networks. Security becomes even harder to achieve in self-organized systems mainly due to problematic key exchange, session handling, absence of centralized services and configuration of all network services on the fly. An efficient key management protocol is an ongoing hot research area. This paper presents the investigation of the problems with the existing key management protocols and Java SE 6 with light weight bouncy castle 1.6 API based implementation of a proposed fully self-organized public key management scheme in mobile ad hoc networks. The scheme allows users to create their key pairs, to issue certificates, to perform authentication and to fully control the security settings of the system without any centralized services.

Keywords: mobile ad hoc networks, self-organization, network security, key management, key authentication, key repository, certificate repository, trust graphs.

1 INTRODUCTION

1.1 Mobile Ad hoc Networks

The term “ad hoc” tends to imply “can take different forms” and “can be mobile, stand alone, or networked”[1]. Ad hoc implies that the network is formed in a spontaneous manner to meet an immediate demand and specific goal. Ad hoc networks have the ability to form “on the fly” and dynamically handle the joining or leaving of nodes in the network. Mobile nodes are autonomous units that are capable of roaming independently. Typical mobile ad hoc wireless nodes are Laptops, PDAs, Pocket PCs, Cellular Phones, Internet Mobil Phones, Palmtops or any other mobile wireless devices. Mobile ad hoc wireless devices are typically lightweight and battery operated.

A mobile ad hoc network (MANET) is an adaptive, self-configurable, self-organizing, infrastructure-less multi-hop wireless network with unpredictable dynamic topologies [2]. By adaptive, self-configurable and self-organizing, means an ad hoc network can be formed, merged together or partitioned into separated networks on the fly depending on the networking needs. i.e. a formed network can be deformed on the fly without the need for any system administration. By infrastructure-less, means an ad hoc network can be promptly deployed without relying on any existing infrastructure such as base stations for wireless cellular networks. By multi-hop wireless, means, in an ad hoc network the routes between end users may consist of multi-hop wireless links. In addition, each

node in a mobile ad hoc network is capable of moving independently and forwarding packets to other nodes.

The domain of applications for ad hoc wireless networks is diverse, ranging from small, static networks that are constrained by power sources, to large-scale, mobile, highly dynamic networks [2]. Such networks are frequently viewed as a key communications technology enabler for network-centric warfare, disaster relief operations, emergency situations, intelligent transportation systems and fault-tolerant mobile sensor grids. Most of these applications demand a secure and reliable communication.

Considering the unique features of ad hoc networks, it is expected that the key management protocols proposed to guarantee the security of conventional networks are not necessarily suitable or adaptable to mobile ad hoc networks. Novel protocols, designed specifically for ad hoc networks, are necessary. Peer-to-peer or pair-wise key management protocols are designed for both fully self-organized mobile ad hoc networks and authority-based mobile ad hoc networks. *Fully self-organized mobile ad hoc networks* do not have any form of online or offline authority [15], [16]. These ad hoc networks are created solely by the end-users in an ad hoc fashion. The users forming the ad hoc networks have no pre-established relationships and, therefore, share no common keying material on their nodes. Users have to set up security associations between themselves, after network formation, without the aid of a common offline trusted third party (TTP). *Authority-based mobile ad hoc networks* support applications that demand the use of an offline authority [16]. In contrast to fully self-organized ad hoc networks, the nodes in authority-based ad hoc networks do have pre-established relationships. The trusted authority sets up the nodes prior to network formation, that is, provides each node with (shared) cryptographic keying material and a set of (universal) system parameters. After network formation, each node becomes its own authority domain and distributes its certificate to nodes within its transmission range.

In the proposed scheme, self-organized behavior is provided by sharing the public keys and self-signed certificates among the nodes to form the network. We use several traditional security mechanisms to encrypt the shared data and the messages. The proposed scheme overcomes the previous pitfalls in getting self-organized behavior, like introducing a trusted node to share the public keys /certificates and sharing a furtive before forming a network.

The self-organized behavior is achieved by forming the network with initial trust phase. The network remains in this trust phase until the trust graph is constructed at each node. In this trust phase, the public keys are shared in a secured environment, and after getting the reply messages for the sent messages of

public keys, the certificates are also shared and wait for replies. In the mean time, the neighbor public keys and certificates are received. After receiving the public keys and certificates of neighbors, each node constructs its key repository and certificate repository. The key repositories will be shared in encrypted form along with certificates. This phase is complete only when all the public keys of nodes are shared the key repositories. Each node will construct its own shared key repository and trust graph. This trust graph will be saved as Master Graph (MG).

The network quits the secure environment when it completes the trust phase because every node will have the public keys of all nodes participated in the network. When each node contains all the public keys of other nodes then nodes can be mobilized. Each node will send the revoked public keys and certificates in encrypted form along with certificates. Each node can become a neighbor to any other node which covers its radio range. Each node offers its certificate encrypted with destination public key. Hence, after sharing the certificate and receiving the neighbor certificate, both will communicate the message in encrypted form along with valid certificate.

1.2 Routing in MANET

The nodes in the network are free to move independently in any direction. Node mobility causes route changes. The nodes themselves are responsible for dynamically discovering other nodes to communicate. The network's wireless topology may change frequently and randomly at unpredictable times. Every node in ad hoc wireless network acts as a router that discovers and maintains routes in the network. Hence, the primary challenge is to establish a correct and efficient route between a pair of nodes and to ensure the correct and timely delivery of packets. Various routing protocols, such as DSDV, OLSR, DSR, AODV, ZRP, etc., have been proposed and widely evaluated for efficient routing of packets [3].

Routing protocols often are very vulnerable to node misbehavior [4]. A node dropping all the packets is considered as malicious node or selfish nodes. A malicious node misbehaves because it intends to damage network functioning. A selfish node does so because it wants to save battery life for its own communication by simply not participating in the routing protocol or by not executing the packet forwarding. A malicious node could falsely advertise very attractive routes and thereby convince other nodes to route their messages via that malicious node.

1.3 Security Attacks in MANET

A security attack is any action that compromises or bypasses the security of information illegally or in an

unauthorized way [2]. The attack may alter, release, or deny data. To secure an ad hoc network, a security protocol must satisfy the attributes: confidentiality (privacy), availability, integrity, authenticity and non-repudiation. *Confidentiality* ensures that secret information in the network is never revealed to unauthorized nodes. *Availability* ensures that the requested network services such as bandwidth and connectivity are available in a timely manner and service is not denied to authorized users. *Integrity* ensures that message or packet being transferred between nodes is not altered or corrupted. *Authentication* ensures the correct identity of the peer node it is communicating with. *Non-repudiation* ensures that the originator of a message cannot falsely deny having sent the message. i.e. if any entity sends a message, the entity cannot deny that the message was sent by it.

The attacks on the ad hoc wireless networks can be broadly classified into two categories: passive attacks and active attacks. A *passive attack* does not disrupt the normal operation of the network; the attacker attempts to retrieve valuable information by listening to traffic channel without proper authorization. Examples of passive attacks are eavesdropping, snooping, information leakage and traffic analysis (traffic monitoring). Passive attacks are very difficult to detect because they do not involve any alteration of the data. One of the solutions to the problem is to use powerful encryption mechanism to encrypt the data being transmitted, thereby, making it impossible for the attacker to get useful information from the data overheard.

An *active attack* attempts to alter or destroy the data being exchanged in the network, thereby, disrupting the normal functioning of the network. Examples of active attacks are: black hole attack, neighbor attack, wormhole attack, denial of service attack, information disclosure attack, rushing attack, jellyfish attack, Byzantine attack, blackmail attack, replay attack, etc. Active attacks can be *internal* or *external*. External attacks are carried out by nodes that do not belong to the network. Internal attacks are from compromised nodes that are part of the network. Since, the attacker is already part of the network, internal attacks are more severe and hard to detect than external attacks. Active attacks, whether carried out by an external adversary or an internal compromised node, involves actions such as impersonation (masquerading or spoofing), modification, fabrication and replication.

1.4 Security Solutions in MANET

Various kinds of security attacks are possible on ad hoc routing. Attack prevention measures can be used as the first line of defense to reduce the possibilities of attacks. There are two types of security solutions: preventive and detective to overcome these attacks. Pre-

ventive solutions are typically based on message encryption techniques, while detective solutions include the application of digital signature and cryptographic hash functions [7], [8]. The prevention schemes proposed for external attacks are key and trust management. The prevention schemes proposed for internal attacks are secure routing protocols.

Message encryption is the science and art of transforming a message into a disguised version which no unauthorized person can read, but which can be recovered in its original form by an intended recipient. The process of encryption and decryption are governed by *keys*, which are small amount of information used by the cryptographic algorithms. There are two types of encryption techniques: symmetric key and asymmetric key. *Symmetric key cryptosystem* uses the same key (the secret key) for encryption and decryption of a message, where as *asymmetric key cryptosystems* use one key (the public key) to encrypt a message and another key (the private key) to decrypt it. Public and private keys are related in such a way that only the public key can be used to encrypt messages and only the corresponding private key can be used for decryption purpose. Even, if attacker comprises a public key, it is virtually impossible to deduce the private key. Symmetric key algorithms are usually faster to execute electronically than the asymmetric key algorithms.

The process of encryption only ensures the confidentiality of the message being sent. *Digital signature* is a technique by which one can achieve the other security goals like message integrity, authentication and non-repudiation. In this, the sender uses a *signing algorithm* and its *private key* to sign the message. The message and the signature are sent to the receiver. The receiver receives the message and the signature and applies the *verifying algorithm* on the message-signature pair. The verification algorithm requires a *verification key*, which is a public key provided by the signer, to verify the document. After verification if the result is true, the message is accepted; otherwise, it is rejected. Hashing can be used for the digital signature process especially when the message is long. In this, the message is passed through an algorithm called *cryptographic hash function* or *one-way hash function* before signing. It is an algorithm which creates a compressed image of the message in the form of a hash value (or message digest) which is usually much smaller than the message and unique to it. Any change to the message will produce a different hash result even when the same hash function is used.

Both digital signature and encryption mechanisms are key-based approaches. *Key distribution and management* is therefore at the center of these mechanisms. There are several methods given in [3] that can be employed to perform this operation, all requiring varying amounts of initial configuration, communication and computation. According to this approach, the key management responsibility is shared among a set

of trusted certification servers called the certification authorities (CAs). Each CA has a public/private key pair, with its public key known to every node, and signs certificates binding public keys to nodes after verifying their authenticity secretly. The trusted CA has to stay on-line to reflect the current bindings, because the bindings could change over time: a public key should be revoked if the owner node is no longer trusted or is out of the network; a node may refresh its key pair periodically to reduce the chance of a successful brute-force attack on its private key.

2 RELATED WORK

The main problem of any public-key based security system is to make each user's public key available to others in such a way that its authenticity is verifiable. In mobile ad hoc networks, this problem becomes even more difficult to solve because of the absence of centralized services and possible network partitions.

In the literature [9]-[13], all or part of the nodes shares some secret to distribute trust among nodes. In this approach, trust or secret for key management should be managed and distributed globally or regionally. This cooperative trust distribution causes inefficiency and vulnerability to active attacks as pointed out in [13], [14].

In the literature [15], [16], trust is managed by each node itself. In the paper [15], two nodes merge their local certificate repository and attempt to find a chain of certificates connecting them. This method is self-organized but its transitivity of trust property is vulnerable to an active attack. In the paper [16], a node directly acquires the public key of another node by handshaking (HS) via a secure side channel. In this method, there is no transitivity-of-trust and no need of priori trust or shared information between nodes from the initial stage. However, HS can be performed only if two nodes are very close; hence, much time is needed to acquire many other nodes' public keys.

In the paper [17], in addition to HS, a certificate request/reply (CRR) procedure that acquires the public key certificates of a remote node via a radio channel is used. In CRR, requests are sent to the nodes that the requester has handshakes before. Returned certificates are verified with the neighbor's public keys that were acquired by HS. If the number of valid certificates containing the same public key is greater than a given threshold, the public key is accepted as valid. This process seems to be cumbersome for maintaining a separate channel to do the handshake process. And there was no chance to encrypt the request and reply messages. In CRR, the certificate verification by the

handshake nodes' public keys enables threshold based certification without any priori shared information or trust, this process results in decrease of throughput. To send a message from a node to another node (non neighbor), sender needs to construct certification path by requesting the existing neighbors.

3 PROPOSED SCHEME

In this paper, we implemented a self-organized public key management scheme by preserving security. The scheme starts the network operations after completing the trust phase with all nodes in secure environment. In the implementation, each device will identify the sensing node as a neighbor and offers its public key. This process will take place only to share the public keys among the neighboring nodes. After successful transfer of the public keys, the certificates are transferred in an encrypted form. As the nodes are in secure environment (i.e. initial trust phase), no need to bother about the intruders. After successful transmission of certificates, the repository transmission starts. Each node needs to remain in trust phase till it acquires all the public keys. After that nodes can be scattered in random order. Each node has to keep the trust phase information for future authentication. Here, a separate channel is avoided to transfer the certificates and the threshold values for authentication by transferring the updated certificates encrypted with previous shared public key. By this approach, we can increase the performance. To provide security, the cryptographic principles take more time to encrypt and decrypt at every node. To avoid this, we use the hybrid encryption techniques that use both the symmetric and asymmetric algorithms.

Each certificate is issued with a limited validity period and therefore contains its issuing and expiration times. After a certificate expires, its issuer issues an updated version of the same certificate which contains an extended expiration time. This updated version is called as the certificate update. Each node periodically issues certificate updates, as long as its owner considers that the user-key bindings contained in these certificates are correct. At each node, these certificates and keys are stored in repositories. The implementation maintains only the neighbor certificates. But, the shared key repositories maintain all the public keys of the nodes in the network. These public keys will be stored along with expire time. At each node, a trust graph (TG) is constructed based on

the shared graphs. This TG will be useful for finding the efficient route. Whenever a change occurs in the shared key repository, immediately it informs the other neighbor nodes about the updated shared key and the trust graph. Based on the trust graph and the public key expire time of each node in the existing path, we find the efficient path for sending the message.

The main offerings of complete self organized public key management scheme include:

1. Any node can join with and leave from the neighbor at any time without informing about the status.
2. Adaptive to survive with rapidly-changing environments.
3. Handling of mobile ad hoc networks with a large number of nodes.

4 PRELIMINARIES

4.1 Network Assumptions

The following are the network assumptions for implementation of the scheme:

- (1) There is no shared initial information between nodes except the IDs of the nodes.
- (2) There is no node that is trustworthy from the beginning.
- (3) The network topology is flexible and the joining and leaving are unrestricted.
- (4) A node that does not properly carry out network protocol such as routing and MAC protocols can be detected by the network monitoring protocol such as in [16].
- (5) All nodes must be connected in trust phase to attain public keys of every node.
- (6) All nodes are evenly distributed and the mobility of all nodes is equal.

4.2 Node Assumptions

- (1) Every node has its own public-private key pairs and will initiate the process by itself.
- (2) Every node will detect the neighbors and by identifying its IP address, offers its public key and waits for the reply message.
- (3) Every node maintains one key repository (KR). Whenever a key with expire time arrived from neighbor's node, will be stored in KR.
- (4) Every node constructs the shared key repository (SKR) for storing public keys of all nodes from the KR. Along with shared key repository, we get a trust graph (TG) of that node. By considering that graph, receiver node will construct its own TG. This graph is represented in adjacency matrix. And for self signed

certificates of neighbors, we maintain certificate repositories (CR). Whenever we get a valid certificate from the neighbor, each node will store certificate in CR.

5 BASIC OPERATIONS

The proposed scheme is mainly based on the applications like rescue operations and military applications. Before executing the military or rescue operations, a network can be formed with all the existing devices and, then, the devices are distributed to all. This means that initially all the devices are in a secure environment and shares the public keys with easy handshake process. After exchanging the public keys of all nodes participated in the network, nodes can be dispersed.

The proposed scheme achieves an extremely self organized behavior by simply giving the security settings of each node to itself. The approach uses trust graphs in place of the certification graphs as proposed in [15], [17]. These graphs are similar to small worlds in PGP. Thus, the trust relationships formed by mobile ad hoc network members must exhibit the same features as PGP system [18].

The major goal of the proposed scheme is to provide a secure environment to send the messages from source to destination. The source will decide the route based on the trust graph which is constructed on its own. The source will encrypt the message that will be decrypted at destination only. However, the source will not allow decrypting the message either by intermediate node or by any other intruder.

The following are the basic operations for implementation of the scheme. Initial phase of the proposed scheme is executed in four steps.

Step 1: Generate the public-private key pairs.

Step 2: Send the generated public key to neighbors, and wait for reply messages.

Step 3: Receive the neighbor public keys and send reply messages.

Step 4: Send the self signed certificates encrypted with neighbor's public keys to those are whom it has received their public keys. Wait for reply message that can be decrypted with its own private key.

Step 5: Receive the certificates from the neighbors, decrypt them and store in certificate repository (CR1) and then, send replies encrypted with their public key.

Step 6: Send the key repository to its neighbor, each node will send encrypted destination certificate and

the repository will be encrypted with public key of destination.

Step 7: Receive the key repository and decrypt it with its own private key and authenticate the message digest with destination public key, update the own shared repository and trust graph with this new update.

Step 8: Send the updated self signed certificate encrypted with the public key of destination and attached with the valid certificate of destination.

Step 9: Send the updated public key to neighbors encrypted with the destination public keys and the digest with the old private key, to allow the destination to decrypt the message with existing public key of sender.

Step 10: Receive the revoked certificates of neighbors and authenticate the message with valid existing certificate and public key.

Step 11: Receive the revoked public key and authenticate the message with existing certificate and public key. And update own shared repository and in turn send the revoked public key and trust graph to their neighbors other than the sender.

Step 12: Compose the message at source with destination address and the required message, encrypt the message with a symmetric key (SK1) generated for that message and the digest with its own private key.

Step 13: Find the efficient route based on the trust graph considering the shortest path and expire times of the router.

Step 14: Encrypt the route and the neighbor certificate with another symmetric key (SK2) and digest with its own private key. And the SK2 will be encrypted with neighbor's public key.

Step 15: Attach the main message composed in Step 12 with the message (route and certificate) composed at Step 14, send it to the neighbor.

Step 16: Receive the message from the neighbor and decrypt the session key with its own private key and get the symmetric key (SK2) and decrypt the message (route and certificate), find whether the destination in the route path is the node itself or other. If the message was supposed to the node itself, then authenticate the certificate and decrypt the main message symmetric key (SK1) with its own private key; otherwise direct the message to its neighbor (based on route) by adding certificate and route path.

We remain in initial phase until we get all the public keys of nodes in a network. After every node is having all the public keys, the trust graph is stored as master graph (MG) and the data should be sent in an

encrypted form. After quitting from the secure environment, we communicate only through the public key encryption and certifications. To authenticate a mobilized node from one group to another group, where the public keys have been changed, we use the MG. Like this, we provide a secure environment with devices whose public keys will be changed periodically. Thus, the proposed scheme security is more compared with other proposals mentioned in the existing system. In our approach, every node trusts its neighbor only through certificates.

A detailed description of each operation is as follows:

5.1 Creation of Public-Private Key Pairs and Public Key Certificates

The public key and the corresponding private key of each user is created locally by the user himself. An expire time is added to the generated public key. A thread is initiated to revoke the public key when it was going to expire at specific time.

Generate an X.509 Certificate based on key pair. This Certificate (Cert) is a self signed certificate that contains serial number, issuer public key, issuing time, expire time with digital signature. The certificate expire time will be less than expire time of the public key, means new certificate will be generated based on the old public key. Whenever the public key is revoked, the new certificate will be generated based on new public key. These certificates are used to provide a session based authentication among the neighbors. More number of certificates are used in between two revocations of public keys.

In the proposed scheme, issuing and revoking certificates are the only operations performed consciously by the users. All the other operations, including authentication, are performed automatically by the nodes, without direct user involvement.

5.2 Exchange of Public Keys and Public Key Certificates

After generating the public keys, each node offers its public key and expire time. This exchange process will be done in secure environment. The public keys are exchanged by easy handshaking process. After getting the neighbor public key, it will store in key repository (KR).

The node receives the public keys from every neighbor and sends reply messages. Every node send its certificate encrypted with destination public key only when it has destination public key and reply

message of its own shared public key. This certificate is encrypted with a symmetric key (SK) generated for the current message. A digest will be encrypted with its own private key. The symmetric key will be encrypted with destination public key. We compose the message like this and send it to every neighbor who has the entry of public key in key repository and the reply message in reply repository. And expects reply message from its neighbor, which will be encrypted as sent message.

5.3 Construction and Exchange of Shared Key Repository and Trust Graph

After getting the neighbor certificates and reply message, each node will move to construct the shared key repository from the obtained public keys and expire times. At present moment, shared repository will have all the public keys of neighbors in key repository (KR). Every node starts the process by constructing its shared key repository from its own key repository. Our main goal is, to wait until we get all public keys of nodes participated in the network. Till that, we wait in initial phase of the network.

Every node offers its own shard repository by filtering both of its primary keys to its neighbors. Means a separate key repository having the updated primary keys are sent to neighbors. In return, if the neighbors have other than these public keys, they will send their shared key repository. Each node receives its neighbor's shared key repository and updates its own shared key repository and informs about the updated information to other neighbors other than sender. Like this, each shared key repository will be updated with public keys of non neighbors.

Along with this shared key repository, each node will construct a trust graph with the maintaining relationships as neighbors. We represent this graph as adjacency matrix and at each node, this graph will be constructed. Each node will send the shared key repository (SKR) and trust graph (TG) in encrypted form.

5.4 Revoking and Exchanging the Certificates and Public Keys

An expire time is assigned to each self signed certificate and a new certificate should be generated with the valid public key. This revoked certificate must be sent to all its neighbors and that message should be sent like a shared key repository, means should be attached with certificate. If any node sends a message with old certificate, then it should be

authenticated. To do this, every node has to save the previous certificate before the revocation. And immediately the node needs to send the new certificate to that neighbor. It is the responsibility of the node itself to send the updated certificates to all its neighbors. We follow the same process for public key revocation.

5.5 Finding a Best Route to Send a Message

The proposed scheme mainly aims to send a message securely from source to destination. Every node already has the trust graph that knows about the topology of the total network. Here, the routing information comes to every node along with shared key repository. At any time, the topology changes so that information will be immediately updated to its neighbor.

The implementation of the proposed scheme makes use of Fisheye State Routing Protocol (FSR). Fisheye State Routing [26] generates accurate routing decisions by taking advantage of the global network information. However, this information is disseminated in a method to reduce overhead control traffic caused by traditional flooding. Instead, it exchanges information about closer nodes more frequently than it does about farther nodes. Hence, each node gets accurate information about neighbors and the detail and accuracy of information decreases as the distance from the node increases.

In FSR, every update message doesn't contain information about all nodes in the network, but in the proposed scheme every update message should contain information about all nodes. At each node, the trust graph is constructed based on the neighbor and shared TG by deleting all the entries (except its neighbors) in its own TG. Each node will not expect the link state message from farther nodes. Instead, information about closer nodes is exchanged more frequently than it is done about farther nodes. The center node has most upto date information about all nodes in the inner circle and the accuracy of information decreases as the distance from node increases.

This procedure of dividing the network into different scope levels is done at each node, meaning that it is independent on a central entity. Even if a node doesn't have accurate information about far away nodes, the packets will be routed correctly because the route information becomes more and more accurate as the packet gets closer to the destination. This means that FSR scales well to large networks as the overhead is controlled.

FSR is suitable for large and highly mobile network environments as it triggers no control messages on link failures. Broken links won't be included in the next link state message exchange. This

means that a change on a link far away does not necessarily cause a change in the routing table. FSR is mainly based on simplicity, as it uses up-to-date shortest routes; it is robust to host mobility as it exchanges partial routing update information with neighbors only, thus reducing routing update traffic. FSR is quality of service (QoS) ready, which means that it is possible to extend the definition of link state by adding bandwidth and channel quality information to the link entry.

It is easy to locate destinations due to the flat addressing scheme and topology map, but this also limits scalability. Other negative points are the routing table storage complexity and the processing overhead. FSR doesn't provide any form of security, as most other protocols.

Once the router has a trust graph and expire times of all node's public keys, it computes the routes based on the Dijkstra's [27] algorithm which computes all shortest paths from a single vertex. The link metric used for path cost is the hop count. If there are more than one equal number of hops then the largest expire time of public keys will be considered. After completion of route procedure, source node obtains the efficient path from source to destination.

5.6 Message Composition at Source

After deciding the destination and the efficient path, the source accepts the message to send it to destination. The source sends the message as same as shared key repository and, additionally, adds the route map to neighbor certificate so that next neighbor will be allowed to decide whether that message is directed to the node itself or to forward. The certificate and route with the neighbor public key is in an encrypted form.

6 EXPERIMENTAL RESULTS

The proposed method, a complete self organized public key management scheme, was coded in Java SE 6 with light weight bouncy castle 1.6 API and run on a Windows XP Professional based Pentium PC 2.4GHz with 512MB RAM. The experimental result shows the proposed method provides both the flexibility and an adaptability of selecting appropriate security configurations to handle dynamic risks in different parts of networks; and also allows a large number of nodes in the network. In the future, we will conduct QualNet 4.5 based simulation study to test the robustness of the proposed scheme.

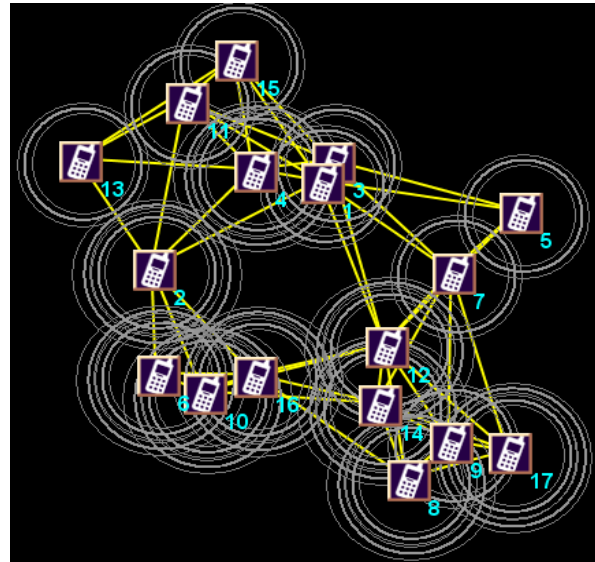


Figure 1 Forming of a mobile ad hoc network

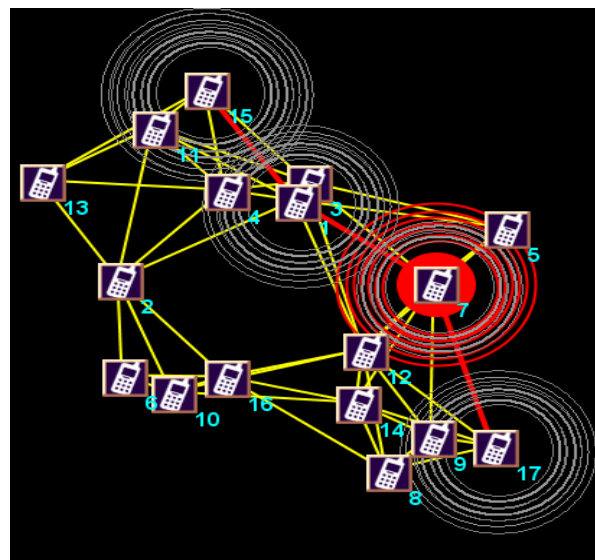


Figure 2 Selection of best route from nodes 15 to 17.

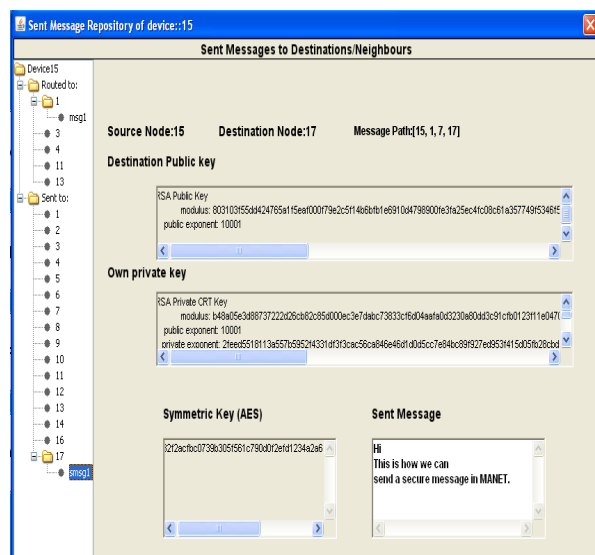


Figure 3 Sent message repositories at node 15.

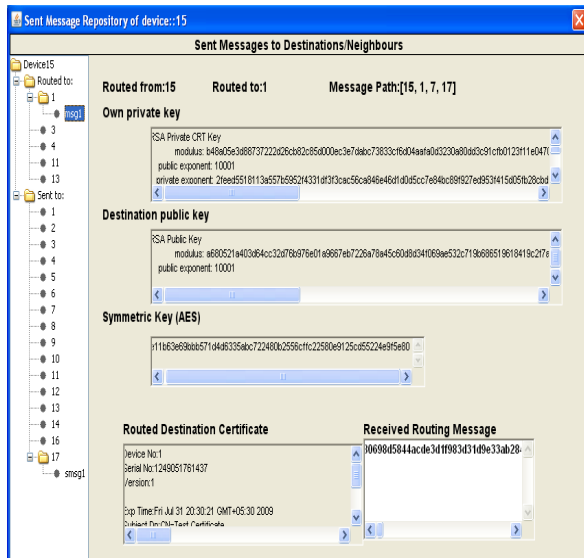


Figure 4 Received message repositories at node 1

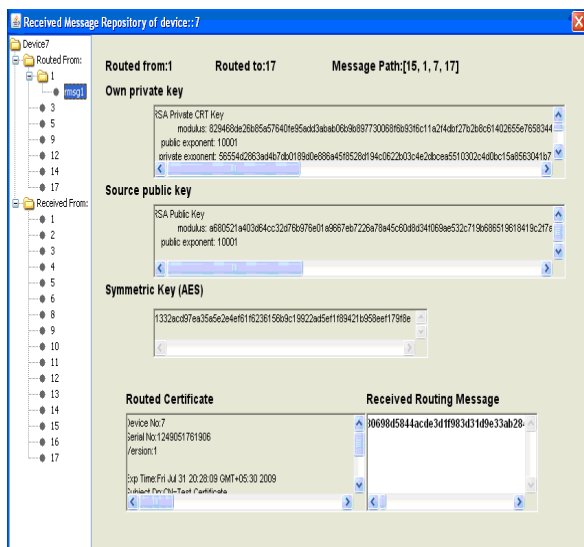


Figure 5 Received message repositories at node 7

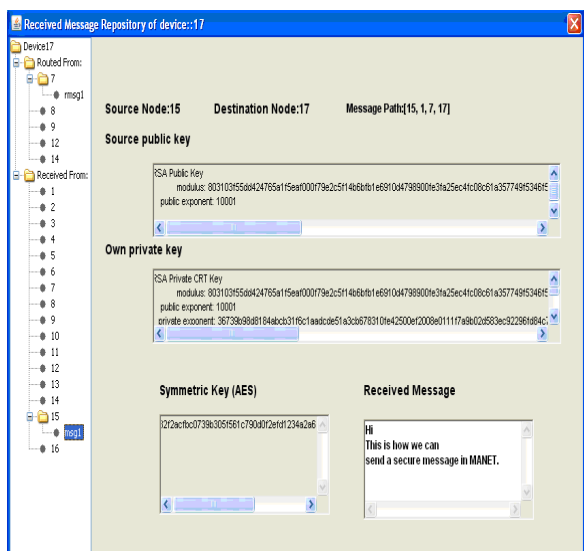


Figure 6 Received message repositories at node 17

7 CONCLUSION

The proposed method, a complete self organized public key management scheme, was coded in Java SE 6 with light weight bouncy castle 1.6 API that allows users to generate their key pairs. It also involves the automatic generation of certificates and performs authentication checking. The work is very useful to fully control the security settings of a given system without any centralized services. Some of the practical applications where the present work finds a lot of significance include key communications technology enabler for network-centric warfare, disaster relief operations, emergency situations, intelligent transportation systems and fault-tolerant mobile sensor grids. In the future, we will conduct QualNet 4.5 based simulation study to test the robustness of the proposed scheme: a complete self organized public key management scheme for mobile ad hoc networks.

8 REFERENCES

- [1] C.K. Tok: *Ad Hoc Mobile Wireless Networks: Protocols and Systems*, Pearson Education, pp. 28-30 (2002).
- [2] X. Cheng, X. Huang and D.Z Du: *Ad Hoc Wireless Networking*, Kluwer Academic Publishers, pp. 319-364(2006)
- [3] C. Siva Ram Murthy and B.S Manoj: *Ad Hoc Wireless Networks: Architectures and Protocols*, Pearson Education (2006).
- [4] F. Anjum and P. Mouchtaris: *Security for Wireless Ad hoc Networks*, John Wiley & Sons (2007).
- [5] Prasant Mohaptra and Srikanth V. Krishnamurthy: *Ad Hoc Networks: Technologies and Protocols*, Springer International Edition (2005).
- [6] C E. Perkins: *Ad Hoc Networks*, Addition Wesley (2001).
- [7] S. Basagni, M. Conti, S. Giordono and I. Stojmenovic: *Mobile Ad Hoc Networks*, IEEE Press Wiley, New York (2003).
- [8] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone: *Handbook of Applied Cryptography*, CRC Press (1996).
- [9] L. Zhou and Z. J. Haas: *Securing Ad Hoc Networks*, IEEE Network Magazine, Vol. 13, No.6, pp. 24-30 (1999).

- [10] H. Luo, J. Kong, P. Zerfox, S. Lu, and L. Zhang: *URSA: Ubiquitous and Robust Access Control for Mobile Ad Hoc Networks*, IEEE/ACM Transactions on Networking, Vol.12, no.6, pp.1049–1063 (2004).
- [11] B. Lehane, L. Doyle, and D. O’Mahony: *Shared RSA Key Generation in a Mobile Ad Hoc Network*, Proceedings of IEEE Military Communications Conference (MILCOM), Vol.2, pp.814–819 (2003).
- [12] B. Zhu, F. Bao, R.H. Deng, M.S. Kankanhalli, and G. Wang: *Efficient and Robust Key Management for Large Mobile Ad Hoc Networks*, Computer Networks- Elsevier, Vol.48, pp.657–682 (2005).
- [13] M. Narasimha, G. Tsudik, and J. Yi: *On the Utility of Distributed Cryptography and P2P and MANETs: The Case of Membership Control*, Proceedings of IEEE International Conference on Network Protocols (ICNP), pp.336–345 (2003).
- [14] S. Jarecki, N. Saxena, and J.H. Yi: *An Attack on the Proactive RSA Signature Scheme in the URSA Ad Hoc Network Access Control Protocol*, Proceedings of ACM Workshop on Security of Ad Hoc and Sensor Networks, pp.1–9 (2004).
- [15] S. Capkun, L. Buttyan, and J. P. Hubaux: *Self-Organized Public-Key Management for Mobile Ad Hoc Networks*, IEEE Transactions on Mobile Computing, Vol.2, No.1, pp.52–64 (2003).
- [16] S. Capkun, J. P. Hubaux, and L. Buttyan, “*Mobile Helps Peer-to-Peer Security*”, IEEE Transactions on Mobile Computing, Vol.5, No.1, pp.43–51 (2006).
- [17] Daeseon CHOI, Younho LEE, Yongsu PARK, Seung-hun JIN, and Hyunsoo YOON: *Efficient and Secure Self-Organized Public Key Management for Mobile Ad Hoc Networks*, IEICE Transactions on Communications, Vol.E91–B, No.11, pp. 3574-3583 (2008).
- [18] S. Capkun, L. Buttyan and J. Hubaux: *Small Worlds in Security Systems: an Analysis of the PGP Certificate Graph*, New Security Paradigms Workshop 2002, Norfolk, Virginia (2002).
- [19] C. Gandhi and M. Dave: *A review of security in mobile ad hoc networks*, IETE Technical Review, pp 335-344, Vol. 23, No. 6 (2006).
- [20] S Marti, T.J. Giuli, K. Lai and M. Baker: *Mitigating Routing Misbehavior in Mobile Ad hoc Networks*, 6th ACM Annual International Conference on Mobile Computing and Networks (MOBICOM 2000), pp. 255-265, Boston, USA (2000).
- [21] P. Papadimitratos and Z. J. Haas: *Secure Routing for Mobile Ad hoc Networks*, Proceedings of SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), pp. 27-31, San Antonio, USA (2002).
- [22] H. Deng, W. Li and D.P. Agrawal: *Routing Security in Wireless Ad hoc Networks*, IEEE Communications Magazine, pp. 70-75 (2002).
- [23] S. Gupta and M. Singhal: *Secure Routing in Mobile Wireless Ad hoc Networks*, Ad Hoc Networks, Vol.1, pp. 151-174 (2003).
- [24] K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields and E. M. Belding-Royer: *A Secure Routing Protocol for Ad hoc Networks*, Proceedings of 10th IEEE International Conference on Network Protocols (ICNP2002), pp. 78-87, Paris, France, (2002).
- [25] L.M. Kornfelder: *Toward a Practical Public-Key Cryptosystem*, Bachelor’s Thesis, Department of Electrical Engineering., Massachusetts Institute of Technology, Cambridge (1978).
- [26] Allen C. Sun: *Design and Implementation of Fisheye Routing Protocol for Mobile Ad Hoc Networks*, Massachusetts Institute of Technology, USA (2000).
- [27] R. Sedgewick: *Weighted Graphs*, Addison-Wesley, chapter 31(1983).
- [28] Internet X.509 Public Key Infrastructure Certificate and CRL Profile - *RFC 2459*.
- [29] Weihong Wang, Ying Zhu, and Baochun Li: *Self-Managed Heterogeneous Certification in Mobile Ad Hoc Networks*, Proceedings of IEEE Vehicular Technology Conference (VTC 2003), Orlando, Florida (2003)
- [30] Matei Ciobanu Morogan, Sead Muftic: *Certificate Management in Ad Hoc Networks*, Symposium on Applications and the Internet Workshops (SAINT’03 Workshops), pp. 337 (2003)