A Hybrid Approach to Key Management for Enhanced Security in Ad Hoc Networks

Aruna Balasubramanian$^{\xi}$, Sumita Mishra* and Ramalingam Sridhar$^{\xi}$**

ξDepartment of Computer Science and Engineering, University at Buffalo, Buffalo, NY 14260-2000
{ab42, rsridhar}@cse.buffalo.edu

*CompSys Technologies Inc, Amherst, NY 14228
mishra@compsystech.com

Abstract

Key management is one of the most challenging tasks in developing security solutions for wireless ad hoc networks. Most of the security primitives can only be realized if an efficient, robust and secure key generation and management system is developed. Symmetric key cryptosystems are difficult to incorporate in an ad hoc decentralized domain, and are not scalable. A public key cryptosystem is computationally expensive and a decentralized certification service is essential for its deployment. In this paper, we present a locally symmetric/globally asymmetric key management solution for wireless ad hoc networks that overcomes the limitations of both. The proposed hybrid approach does not require prior trust assumptions or the presence of a centralized certification authority. Preliminary analysis and results indicate that this approach can achieve a better performance when compared with the traditional solutions. We show that our approach is scalable, and robust with respect to node failures, topology changes, node speed and loss of connectivity.

Keywords

Ad hoc networks, Security, Key management, Symmetric cryptosystems, Asymmetric cryptosystems, Threshold cryptography, Clustering

I. Introduction

Mobile ad hoc networks (MANETs) are formed among a set of devices that communicate using the wireless radio link and are provided with minimum infrastructure support. The adoption of this technology is becoming widespread at homes, industry and government organizations largely owing to their ease of use, flexibility and low installation costs. One of the most profound impacts of wireless ad hoc technology is in mission critical applications. It is indispensable in the military, national defense and rescue/relief operations, where wires and other infrastructure cannot be guaranteed and networks need to be formed according to convenience. Nevertheless, there is a growing concern in using this technology because of its security vulnerabilities.

Use of wireless links renders an ad hoc network susceptible to link attacks ranging from passive eavesdropping to masquerading, message replay, and message distortion. Nodes roam in a hostile environment with relatively poor physical protection and can be compromised and impersonated. Key generation and distribution is one of the most challenging tasks faced by the security protocol designers in this domain. Traditional key management solutions require online trusted authorities or certificate repositories and are not well-suited for ad hoc networks. Key management solutions for ad hoc networks also needs to adapt to large network sizes, frequent loss of connectivity and often needs to cater to low-power battery operated mobile devices. To the best of our knowledge there is no comprehensive key management solution in this domain and most existing ones have severe limitations [1].

This paper describes our solution to the key management problem in ad hoc networks. The goal is to provide a robust, decentralized, scalable and secure key management infrastructure and is designed to reduce computational complexity. To achieve this, we propose a hybrid (symmetric and asymmetric key based) key generation scheme. While a symmetric cryptosystem is suitable for low power devices, it is not scalable, and while asymmetric cryptosystems are more secure and scalable, they are computationally intensive. We propose a locally symmetric and globally asymmetric key based solution that overcomes the limitations of

** Contact author

both. Localization is achieved by grouping nodes into clusters. There is no assumption of pre-distributed symmetric keys or an external certification authority.

To this end, we have designed a novel key generation algorithm for simultaneously generating pair-wise secret keys and (public, private) key pairs for each node in the cluster in a low cost initialization phase. The uniqueness of this design is that, the pair-wise secret keys for symmetric encryption are not explicitly exchanged but are calculated by each node independently from a partial key. This not only allows a decentralized key exchange protocol, but also reduces the volume of message exchange and thus improves performance. Decentralization of the certification authority is achieved using threshold cryptography [2] [3].

Though the concept of threshold cryptography and secret sharing have been used for key management [4], [5], we present a unique approach that provides significant enhancement over available solutions. Recently, a cluster-based scheme was proposed for key management in ad hoc networks, using threshold cryptography for decentralization [6]. Our solution is significantly different from this approach with respect to the cluster-management scheme and the use of the hybrid protocol.

A simulation model has been developed and an extensive evaluation is presented. The analysis indicates that the security levels achieved for ad hoc networks is significantly enhanced using a hybrid symmetric/asymmetric cryptosystem when compared with traditional approaches. The domain of this work is ad hoc networks; but we believe that the approach can be easily adapted to other wireless networks and in particular sensor networks, and even possibly wired networks (which needs further study).

The paper is organized as follows - The next section presents related research and in section 3 we detail the concepts used in the solution. In section 4 we describe the hybrid security solution Section 5 presents the evaluation of the security architecture both by analysis and simulation. We simulated ad hoc networks that use our architecture, for proof of concept and for studying the performance and overhead. The conclusions and future work are presented in section 6.

II. Security in ad hoc networks

Recently several research efforts have addressed the issue of key management for ad hoc networks. Most of these efforts focus on solutions based on asymmetric techniques, while symmetric cryptosystems are studied more in the context of sensor networks [7], [8]. Recently, a distributed symmetric key management solution for ad hoc networks was proposed [9]. This solution is however, more suitable for nodes that are within each others range, rather than multi-hop nodes. Key management services using asymmetric cryptosystems, usually seek to establish a public key infrastructure or a certification service within the constraints of the ad hoc network domain.

Zhou and Haas [4] used the concept of threshold cryptography for distributing certification services. The private key of the certification authority (CA) is distributed among a group of special servers and node requests these servers to provide distributed certification services. However, the solution does not address issues of availability and connectivity to the special servers. Also, use of special servers may not always be possible in an ad hoc network.

Along similar lines, Yi and Kravets, provides a flexible, decentralized and mobile certification solution by distributing the CA functionality to selected nodes based on security and physical characteristics of the node [10]. This work addresses issues of availability and connectivity to the mobile agents, but are not truly decentralized. Kong, *et al* provide a more decentralized solution where every node stores a key share of the certification authority's signing key. Certification services are provided by one-hop neighbors that can sign partial certificates [5]. However, restricting certification authority to one-hop neighborhood may affect the availability. Increasing the availability, by decreasing the required number of partial signatures, would render the system susceptible to attacks.

Identity-based cryptosystems [11], are based on binding the public key of a node to its identity. Several solutions have been proposed for distributed key management using identity-based systems [12]. An important limitations in having a unique private key mapped to the identity of a node is that, refreshing the keys in case of node compromise is difficult. Recently a cluster-based scheme was proposed for securing ad hoc networks using cluster heads as the secret share agents and the secret shares are distributed using threshold cryptography [6]. However, connectivity and availability issues have to be addressed if nodes need to contact a certain number of cluster heads for certification services. Cluster heads may also be vulnerable to single point of attack.

Hubaux *et al* [13], proposed a public key infrastructure based on certificate graphs such as that used in PGP [14], where nodes store a subset of the certificate graph in their repository. Two nodes (A and B), that need to obtain each other's certificate merge their repositories and get a full certificate chain from A to B. The primary limitation is the time taken and the complexity for initialization and formation of the certificate graphs. Also, in sparse networks, there may not be enough trust relationships to form a certificate chain when two repositories are combined. Asokan and Ginzboog extended two-party Diffie-Hellman key exchange to multi-party [15]. Similarly there are several group key agreement schemes [16], [17] for securing group communication such as secure broadcast, but these cannot be adapted to two-party communication.

As can be seen, most of the available solutions for key management either need special servers, are not scalable, assume prior trust relationships, or are vulnerable to attacks. There is a need for a comprehensive solution that addresses these issues and is at the same time low in complexity. In our work, security solutions are looked at from a hybrid perspective and the adoption of both symmetric and asymmetric key cryptosystems is introduced. The proposed solution is well suited for ad hoc networks, and addresses the concerns raised above.

III. Design concepts

In our hybrid key management solution, nodes are divided into non-overlapping clusters and local intra-cluster communication is secured using symmetric key cryptosystems while asymmetric cryptosystem is used to secure global inter-cluster communication. The scalability issues of symmetric key encryption related to exchanging keys with all nodes in the network, is addressed by restricting the participating entities to within a cluster. On the other hand, assymetric cryptosystem is used only for inter-cluster communication, reducing the number of messages encrypted using asymmetric keys, thus reducing the complexity of the operation. To build a decentralized solution, we have designed a distributed certification authority using threshold cryptography for asymmetric cryptosystems and a decentralized key distribution mechanism for exchanging symmetric keys.

In this section, we will give a brief overview of the design concepts.

A. Threshold cryptography

The security of an asymmetric key cryptosystem, largely depend on reliably communicating a nodes public key to others. One of the key threats in obtaining the authentic public key of a node is the intruder-in-the-middle attack [13]. To solve this problem a certification authority (CA) is used, that certifies the authenticity of a node and reliably provides its public key. Traditionally, the certification authority is a central server or a trusted third party server centrally located.

To achieve decentralization of the CA we use a threshold cryptography scheme where the certification service is distributed to the nodes in a *k-out-of-n* manner. This is achieved by distributing a secret share to all n nodes, such that any k nodes when combined can regenerate the certification authority's signing key, but no combination of $k-1$ shares can recreate the key. The scheme is ideal for a dynamic network where a temporary loss of connectivity is common. A genuine user needs to contact only k of the n nodes to obtain certification services. However, an adversary has to corrupt at least k nodes to be able to masquerade as the certification authority. When Byzantine conditions are assumed [18], the threshold cryptography scheme is secure when $k \leq n/2$.

B. Cluster-based approach

Clustering increases scalability by restricting information sharing and computations to within a cluster. In this solution, we also use clustering to provide localized services.

Two most common cluster architectures are : Hierarchical and Flat. In an hierarchical architecture special nodes called cluster heads are delegated authority to manage the other nodes in the network. In a flat architecture, all nodes are considered equal and all management protocols are truly decentralized. While using hierarchical clustering, special algorithms need to be designed for choosing the cluster heads and these cluster heads are vulnerable to single point of attack or failure. In this solution we advocate a flat architecture where the cluster head is nominal with very few additional responsibilities and any node in the cluster can be designated as the head. Apart from ensuring that the solution is decentralized, this design also helps in reducing the risk of single point of attack/failure.

IV. Hybrid approach to key management

The protocol can be divided into three conceptual tasks - a) Clustering and key generation b) Determining keys for secure symmetric and asymmetric communication c) Building a distributed certificate authority

A. Clustering and Key generation

A.1 Clustering

Cluster-based solutions have been used for several applications, such as reliable routing [19], power control [20] and others, such as the one proposed in [21]. Any of the available clustering schemes can be used or clusters can be formed as needed. In this work, the solution architecture is not affected greatly by the algorithm used for clustering, but is only affected by the number of nodes in each cluster. As will be detailed in the next section, any node can take up the role of the cluster head with only minimal additional responsibility. Inter-cluster communication is seamless and proceeds as though there are no clusters.

Clustering schemes particularly suited for our proposed solution is currently being investigated.

A.2 Protocol for key generation

The key generation protocol is adapted from the GDH.2 algorithm [22], and extends Diffie-Hellman key exchange to n parties. There are several protocols that adopt this algorithm for generating keys [16] and for authenticated key exchange [23] used for securing group communication. In this paper, we have designed a GDH.2 based key generation algorithm for point-to-point communication. Though, both symmetric and asymmetric keys are generated for each node, it can be shown that the complexity of generating both keys is much less than the sum of their complexities. In fact, in certain cases, it can be shown that the complexity is less than that of generating only symmetric or only asymmetric keys.

Set Up:

The following notations are used throughout the protocol description,

G	Unique subgroup of Z_p^* of order q with primes p and q
q	Order of the algebraic group
α	Exponentiation base, generator in group G
n	Total number of nodes in the cluster
N_i	Secret exponent generated by the i^{th} member of the cluster
Z_i	Message exchanged between i^{th} and the $(i+1)^{th}$ element of the cluster
$C_{i,k}$	Partial key generated by node i in the k^{th} round

All arithmetic is performed in the cyclic group G of prime order q, which is the subgroup of Z_p^* for a prime p such as $p = kq + 1$ for some small $k \leq n$. The security of the algorithm is based on the difficulty of solving discrete logarithm problems in a subgroup with this setting [24].

The basic cryptographic property used in this algorithm is that, any message M can be written as

$$M = \alpha^{C} \ mod \ p \tag{1}$$

for a primitive root α of prime p, and C is the discrete logarithm of M for $(\alpha \ mod \ p)$. An adversary having the values of (M, α, p) cannot deduce the value of C because of the difficulty of computing discrete logarithms [24]. By the cyclic group property, if the inverse of an integer K is $K^{-1} \ mod \ q$,

$$J^{K*R1*K^{-1}} \ mod \ p = J^{R1} \tag{2}$$

Another important result is the Euler's theorem, which states that

$$X^{\phi(N)} = 1 \ mod \ N, for \ all \ integers \ N, \ when \ gcd(X, \ N) \ = \ 1 \tag{3}$$

where ϕ is Euler's totient function [24].

Key generation:

The key generation protocol is implemented in n rounds where in the first *n-1* rounds, the participating entities contribute secret keys. In the final round, the n^{th} node creates the partial keys from the individual contributions and multicasts them to the other cluster nodes. The key generation uses contributory group key agreement principles, where every node contributes to the formation of the key.

The nodes in the cluster are assumed to order themselves and assign numbers from 1 to n such that an entity *Node i* would know the entities with assignments *Node i+1* and *Node i-1*. The set up variables *G, p, and q* are generated by *Node 1* and used only by *Node 1* and thus, the variables need not to be exchanged with the other nodes in the cluster. α is used by Nodes 1 and 2 and is communicated from *Node 1* to *Node 2*.

The initialization phase of the protocol is described in Fig. 1.

Round i

1. *Node i* selects a secret key N_i such that, $N_i \in Z_q^*$
2. *Node i* to *Node i+1*,

$$\textit{Node i} \xrightarrow{M_i=\alpha^{N_1*N_2...N_i},\, C_{i,k}} \textit{Node i+1}$$
$$C_{i,k} = \alpha^{(N_1*N_2*..N_i)/N_k} \, \forall k \in [1, i] \tag{4}$$

Fig. 1. Key generation protocol: Initialization phase (n-1 rounds)

Node 1 initiates the key generation algorithm by generating a secret key N_1 and communicating to *Node 2*. The first message, sent from *Node 1* to *Node 2* is,

$$\textit{Node 1} \xrightarrow{M_1=\alpha^{N_1},\alpha} \textit{Node 2} \tag{5}$$

M_i, (refer Fig .1) represents the intermediate key generated by nodes in *step i* and $C_{i,k}$ is the intermediate partial key of *Node k* in *step i*. The partial contribution of a node, is the total contribution (M_i) without its own contribution. M_i and $C_{i,k}$ are computed using the input from *step i-1* for $i \geq 2$ as follows,

$$M_i = (M_{i-1})^{N_i} \tag{6}$$
$$C_{i,k} = (C_{i-1,k})^{N_i} \forall k \in (1, i-1) \tag{7}$$
$$C_{i,i} = M_{i-1} \tag{8}$$

By building up these contributions, the last node in the cluster, *Node n*, receives the message,

$$M_{n-1} = \alpha^{N_1*N_2...N_{n-1}}$$
$$C_{n-1,k} = \alpha^{(N_1*N_2*..N_{n-1})/N_k} \, \forall k \in (1, n-1) \tag{9}$$

The n^{th} round is implemented by *Node n* using the following algorithm.

Round n:

1. *Node n* generates a secret key N_n such that, $N_n \in Z_q^*$
2. *Node n* calculates partial keys as

$$C_{n,k} = \alpha^{(N_1*N_2*..N_n)/N_k} \, \forall k \in (1, n) \tag{10}$$

and multicasts the key to the other nodes in the cluster.

3. $Node\ i\ \forall\ i \in [1, n]$ create their own (private, public) key pair using $C_{n,i}$ as the public key, such that the private key PV_i satisfies the equation

$$(C_{n,i})^{PV_i} = 1\ mod\ \phi(P_i * Q_i) \tag{11}$$

where P_i and Q_i are large primes generated by *Node i*. Equation. 11 ensures that the (private, public) key pair can be used for assymetric key cryptosystems and will elaborated in the next section.
4. All nodes publish the value of P_i*Q_i, $\forall i \in [1, n]$ as one of the public key pairs.

The message exchanges during key generation is illustrated in Fig. 2. Keys are refreshed periodically, to ensure security.

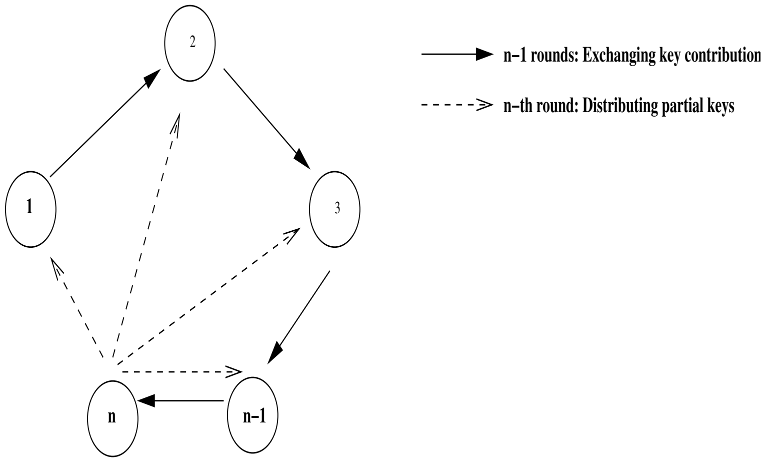
n−1 rounds: Exchanging key contribution
n−th round: Distributing partial keys
1
2
3
n
n−1

Fig. 2. Illustration of the key generation algorithm

B. Determining keys for symmetric and asymmetric cryptosystems

The partial keys generated during initialization are used to determine the symmetric and asymmetric keys for securing communication.

B.1 Asymmetric key generation

To design an asymmetric key cryptosystem, a (public key, private key) pair is generated such that any message encrypted using one of the keys can only be decrypted using the corresponding key pair. In our solution, we have determined the public key of *Node k* to be of the form, $(C_{n,k}, L)$, where $L = \phi(P_k, Q_k)$.

$C_{n,k}$ is the partial key of *Node k* that has been multicast during key distribution as shown in Fig. 2. The private key corresponding to this public key is PV_i and known only to *Node k*.

The encryption/decryption algorithm used is adapted from RSA [25] and the public and private key pair are determined so that the same algorithm used for encryption and decryption in RSA, can be used in this case as well. To show this, consider a message $M \leq P_i * Q_i$ encrypted by *Node i* using the private key as

$$Encryption(M) = M^{PV_i}\ mod\ (P_i * Q_i) = D \tag{12}$$

The crypto-message D is decrypted as follows

$$\begin{aligned} Decryption(D) &= D^{Public\ key}\ mod\ (P_i * Q_i) \\ &= D^{C_{n,i}}\ mod\ (P_i * Q_i) \\ &= M^{PV_i * C_{n,i}}\ mod\ (P_i * Q_i) \\ &= M^{(k*\phi(P_i*Q_i))+1}\ mod\ (P_i * Q_i) (from\ Eqn.\ 11), \\ &= M * M^{k*\phi(P_i*Q_i)}\ mod(P_i * Q_i) \\ &= M (from\ Euler's\ theorem\ Eqn.\ 3) \end{aligned} \tag{13}$$

Usually, both the private and the public key is determined by the node itself. Each node then publishes it's public key and requests for a certificate binding the node and its respective public key. In the absence of an external authentication agent such as a central server, it may be difficult to verify the validity of the published public keys. This situation creates a potentially security hazard. However, in this solution, the public key is already determined and multicast to all other nodes in the cluster, and the private key is chosen accordingly. Because of this, a malicious user, cannot spoof a Node (say A), by publishing a false public key of A and obtaining the certificate for this wrong key. Additionally, *Node n*, cannot publish the wrong public key of *Node A* during multicast, because *Node A* can verify the validity of its own public key.

The security of the scheme is based on the difficulty of factoring and has been discussed in detail in [25]. Strong prime numbers need to be chosen to ensure the security of the solution. Security is also determined by the length of the secret key and the key refresh time.

B.2 Symmetric key generation

In symmetric key cryptosystems, a secret key is shared between each pair of nodes and pair-wise communication is secured using this key. Our algorithm is unique in that no pair-wise key exchange is needed for generating the secret keys. Instead, the partial keys are used by a node individually, to create pair-wise keys with each node in the cluster. In order to use this algorithm for symmetric key encryption, the following statement needs to be proved.

Statement 2. *If Node i and Node j individually generate a secret key to secure their pair-wise communication, secret key generated by Node i = secret key generated by Node j, and any other node cannot generate the same key.*

Proof. Assume *Nodes i and j* in a cluster need to establish a secure communication. *Node i* and all other nodes in the cluster know the public key of *Node j*, which is $\alpha^{(N_1 * N_2 .. N_n)/N_j}$. *Node i* knows its own secret key N_i and this is not known to any other node in the network. It calculates the shared secret as,

$$\begin{aligned} \textit{Shared Secret (Node i)} &= \textit{(Public key of Node j)}^{\,N_i^{-1}} \\ &= \alpha^{(N_1 * \ldots N_n/(N_j) * (N_i^{-1})} \\ &= \alpha^{(N_1 * \ldots N_n)/(N_i * N_j)} \textit{(from Eqn. 2)} \end{aligned}$$

Similarly, *Node j* knows the public key of *Node i*, which is $\alpha^{(N_1 * N_2 .. N_n)/N_i}$.

$$\begin{aligned} \textit{Shared Secret (Node j)} &= \textit{(Public key of Node i)}^{\,N_j^{-1}} \\ &= \alpha^{(N_1 * \ldots N_n/(N_i) * (N_j^{-1})} \\ &= \alpha^{(N_1 * \ldots N_n)/(N_i * N_j)} \textit{(from Eqn. 2)} \\ &= \textit{Shared Secret (Node i)} \end{aligned}$$

Hence *Nodes i and j* have created the same shared secret key even though pair-wise keys are not exchanged explicitly. It has been proven that N_i^{-1} and N_j^{-1} cannot be calculated without the knowledge of N_i and N_j [23]. This in turn proves that the pair wise secret key cannot be computed by an intruder or any other node, without the knowledge of N_i or N_j and thus is secure.

C. Distributed certification service.

In this work, we have designed a distributed certification authority to securely obtain the public key of an entity. For decentralization, threshold cryptography is used, where the private key of the certification service is distributed in a *k-out-of-n* manner to all nodes in the network. Any k of these nodes can combine their secret shares to form the private key, while the public key of the service is common knowledge. The maximum number of malicious users is assumed to be $k-1$ and no combination of $k-1$ partial secret keys can regenerate the private key of the certificate authority. The algorithm for certificate distribution is presented in Fig. 3.

Certificate generation
1. *Node A* requests the cluster nodes for a certificate validating its public key.
2. The nodes in the cluster that believe in the authenticity of *Node A*, reply with a partial certificate.
3. The partial certificate contains, among other things, the identity of the requesting node, and the public key of A signed using the secret key share.
4. *Node A* collects k such certificate shares and combines them to form the complete certificate.

Certificate verification
1. A checks the validity of the certificate.
2. If the combined certificate is not valid, A uses a different combination of certificate shares to generate the complete certificate.

Fig. 3. Certification service.

Note that any node in the network can assume the role of a distributed certification authority because of its ability to create partial signatures. In this respect, our solution is similar in concept to that presented in [5]. However, we do not restrict the service providing capabilities to one-hop neighbors. Instead, localized certification services are provided by the cluster nodes.

An entity signs a partial certificate only if it is convinced of the authenticity of a node. We suggest the use of some out-of-bound physical proof or the use of location-limited side channel for authentication [26]. When new nodes join the network, they can form a new cluster, use an auxiliary key agreement methodology [16] to generate a new pair of keys in an existing cluster or wait for the next key refresh interval.

Certificate validity: A certificate is valid only for a time period and is specified when issued. The certificate expires after this time and a new certificate request needs to be generated. If a node moves and joins a new cluster in this time period, the certificate can either be renewed, or a new pair of keys can be generated.

Certificate revocation: A certificate can be revoked on their expiry or evidence of bad behavior. When a certificate is revoked due to misbehavior, the node id is maintained in a certificate revocation list (CRL) and the CRL is flooded to all the nodes in the network. The certificate expiry information is available along with the certificate, and can be verified easily. All nodes also check the CRL when presented with a certificate, to check its validity.

Certificate renewal: A certificate is renewed either if it expires or if the cryptographic keys are refreshed. To renew a certificate, an entity presents its old certificate, and the local certification service checks the validity of the certificate and the CRL to issue a new certificate. If a node moves to a new cluster, the authenticity of the node needs to be proved again by some out-of-bound mechanism for issuing certificates. If certificates are renewed due to key refresh, the certificate generation process is repeated.

Secret share update: The secret shares of each nodes need to be updated periodically, to ensure that a mobile adversary does not compromise k users over a period of time. If such a compromise occurs, the private key of the certification authority is no longer a secret and spurious certificates can be signed by the adversary. By updating the secret share of each entity periodically, a malicious node needs to compromise k users within the period to be able to obtain the private key. Secret shares are updated such that the total secret remains the same. There has been several methodologies proposed for secret share updates [5], [4], [27] and we will adopt one such methodology. Currently, we are investigating secret share update schemes most suitable for our approach. Another desirable property is verifiable secret shares (VSS), in which signing certificates with a wrong secret share can be detected publicly [28]. Some pro-active secret share update schemes, also discuss the use of verifiable secret shares [27] and our scheme can be enhanced by adopting the VSS technique.

V. Evaluation

A. Protocol Analysis

The goals of the security solution are security, scalability, robustness and low computational cost. In this section, we will present an analysis to show that the protocol achieves the objectives mentioned.

A.1 Security

To ensure that the designed solution is secure, the different stages of the protocol is analyzed.

Key generation: Key generation involves exchanging partial contributions (alternatively, the secret key) among nodes. The secrecy of the partial contributions is ensured by the multi-party Diffie-Hellman key exchange mechanism used [22]. The intermediate secret key and the intermediate partial keys cannot be used to construct the secret keys of the nodes and thus they are not susceptible to eavesdropping attacks.

Key distribution: In the n^{th} round of the protocol, the partial keys are distributed to all the nodes in the network and any eavesdropper, tuned to the transmission frequency, can spoof the information exchange. However, the partial keys cannot be used to construct the secret key and hence the key distribution mechanism is not a potential security risk [22].

Security of symmetric encryption: The security of the symmetric key encryption is based on the secrecy of the pair-wise secret key. To determine the pair-wise secret key of two nodes, A and B, the inverse of the secret keys, namely N_A^{-1} or N_B^{-1}, need to be known. It has been shown that the inverse cannot be calculated without the knowledge of the secret keys N_A and N_B that are known only to A and B, as discussed in earlier. Thus, the symmetric key cannot be constructed by a malicious node. Additionally, the adoption of the contributory key agreement protocol GDH.2, provides resistance to known-key attacks and offers perfect forward secrecy [23].

Security of asymmetric encryption: The asymmetric encryption algorithm used in this solution is based on RSA and is very secure against cryptanalysis, because of the difficulty in factoring. Thus, the private key used for asymmetric encryption cannot be broken and this has been analyzed in detail [25]. Another potential security risk while using asymmetric key encryption is in obtaining the public key of the communicating entities. The distributed certification authority designed in this solution ensures that the public keys can be reliably obtained. This solution is tolerant to a maximum of $k - 1$ malicious nodes.

Non-Vulnerability of cluster heads: Traditional solutions using a cluster-based approach delegate authority to cluster heads, making them vulnerable to a single point of attack. This is because, if a single node can control the network, capturing the node is similar to capturing the network. However, in this work, secret key of nodes cannot be obtained by compromising the cluster head, as the secret keys of nodes are neither created or known to the these nodes. The role of the cluster head is restricted to distributing partial keys, and this role is transferable to any other node in the cluster. Thus, if the cluster head breaks down during key generation, this role can be taken up by the $(n-1)^{th}$ node. Another potential security risk is the leakage of the value of the prime numbers (P and Q), in the event of a cluster head compromise since the security of the RSA scheme depends on the security of the prime numbers. To maintain the secrecy of P and Q, they are created at random, used during the formation of the partial keys and then destroyed.

Node compromise: If a node A is compromised, the secret key of A is known to the intruder, and a masquerading attack can be launched. However, secret key of A cannot be used to obtain the secret keys of other nodes in the cluster. Additionally, we assume that the compromised node can be tracked using misbehavior tracking mechanisms [29].

A.2 Scalability

Grouping nodes into non-overlapping clusters and generating keys locally within the clusters, help in adapting the solution to large network sizes. The certification services are also provided locally, and an increase in network size will not affect availability or connectivity to the certification authority. Also, symmetric key encryption is restricted to intra-cluster communication, reducing the number of participating entities. This addresses the scalability issues faced by conventional symmetric key cryptosystems.

A.3 Robustness

The solution is robust against node failures and loss of connectivity. Node failures during initialization of keys do not affect the key generation methodology. We assume that nodes can rearrange their order (from 1 to n) in the event of an intermediate node failure before exchanging its contribution. The solution is also not affected by the failure of the cluster head (or Node n) during key generation. The role of the cluster head can be taken up by any node, and the previous node (n-1) can create and distribute the partial keys in the event of cluster head failure. Node failures or compromise after key generation will not affect the working of the security solution because there is no dependency between nodes for secure communication. Certification services are provided by any k entities in the cluster, and thus the solution is tolerant to node failures and loss of connectivity as long as there are enough nodes to provide the service.

A.4 Computational cost

Symmetric key cryptosystems are computationally much less expensive than asymmetric key systems. However, they are usually not deployed for ad hoc networks because a decentralized key management scheme for exchanging symmetric keys is difficult to implement and the solution is often not scalable. By using the hybrid scheme, we have a designed a scalable symmetric key exchange methodology for use within a cluster. By encrypting inter-cluster communication using symmetric key encryption the computational cost of using asymmetric system for inter-cluster communication as well is saved. It should be noted that the cost of the hybrid approach is less than the sum of the cost of the symmetric and asymmetric approaches, when applied individually. This is because, the partial keys used for determining both the symmetric and the asymmetric keys are generated in a single round.

B. Simulation

In order to evaluate our concept, we have developed a simulation model of our protocol using GlomoSim [30]. The performance of the proposed approach is tested for different scenarios and is compared with traditional solutions. The simulation model is also used to study the delay characteristics of the protocol for varying levels of mobility and network sizes.

B.1 Simulation environment

The protocol is defined at the application-layer and we use UDP packets for message transfer. The simulation area is $1000m$ X $1000m$, and the number of nodes in the network vary from *30* to *100*. We study the performance of the protocol for varying speeds from $1m/s$ to $20m/s$ and the random waypoint model is used for node mobility. We assume a standard bit error rate, as provided by GlomoSim and 802.11 protocol is used at the MAC layer. Nodes form clusters and create and distribute keys using the contributory key generation scheme. All nodes then request for certificates using a broadcast message, and obtain certificates from the cluster nodes. The total simulation time is *20 minutes* and certificates expire between *300* to *375s*, chosen randomly. Nodes are assumed to request for new certificates between *20* to *30 seconds* after the certificate expires. These parameters are chosen so that a reasonable number of certificate requests will be generated during the simulation time.

The number of malicious nodes are assumed to be 1/5 of the total number of nodes and are chosen randomly and distributed uniformly in clusters. A valid certificate can be obtained only if it is signed by k nodes, where $k-1$ is the number of malicious nodes in a cluster. The social factors, such as a location-limited channel, involved in obtaining the certificates is ignored. We assume that malicious nodes do not provide certificates and all other nodes respond to the certificate request query with a probability of 0.9.

Retransmissions and acknowledgments are built over the unreliable UDP, to provide for reliable packet delivery.

B.2 Comparison study

The hybrid and the centralized approach are simulated and the performance is compared under similar settings for varying mobility.

We have determined the following metrics to study the performance of the solution: Success ratio and Delay per certificate. Success ratio is defined as

Number of successful certificate requests / Total number of certificate requests

While the success ratio determines the effectiveness of the solution, the cost is evaluated using the delay. Delay per certificate is defined as

Total delay for certificate generation / Number of certificates signed

For the hybrid architecture, the latency includes the key generation and distribution time, while for a centralized approach, the delay solely consists of the latency for certificate generation. The simulation was performed for a network size of 50 nodes.

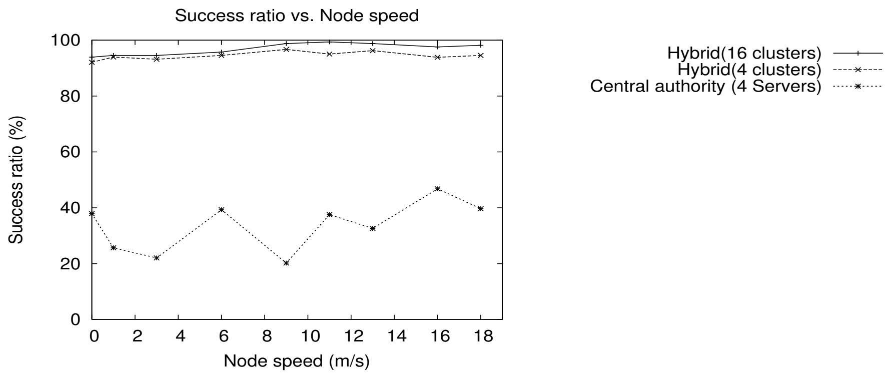

Fig. 4. Comparison of success ratio

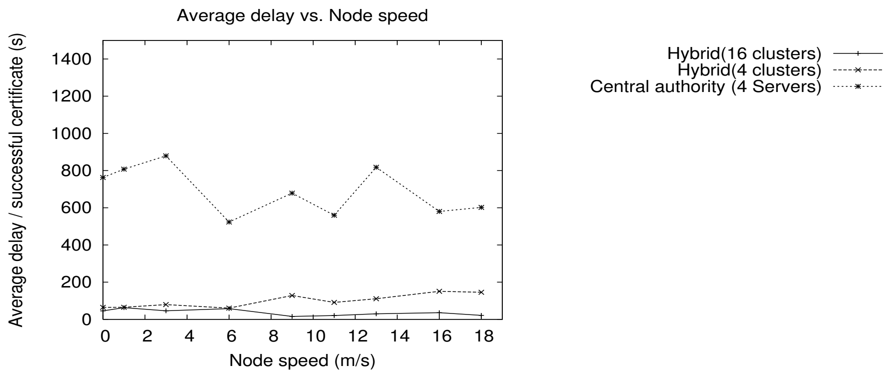

Fig. 5. Comparison of average delay

It can be seen from Fig. 4, that the success ratio of the hybrid approach is significantly higher than that of the centralized approach for varying node mobility, even when four central servers are used for providing certification services. This confirms that the centralized approach is not suitable for ad hoc networks, where connectivity to the central server cannot be guaranteed. By clustering nodes and providing localized certification service, connectivity to the distributed CA is ensured. The hybrid approach is also robust in terms of mobility and even at high node speeds of $20m/s$, the success ratio is close to *100 %* when the number of clusters is *16*.

The efficiency of our protocol is demonstrated in Fig. 5, where it can be seen that the average delay to obtain a certificate is less than *50 seconds* and the solution is robust with increasing mobility. The total delay includes the timeouts and other overheads incurred even if a certificate request fails, but is averaged over only the number of signed certificates. The average delay of the centralized approach is prohibitively high because of the large timeouts and small number of successful certificate generation.

We observe from Fig. 4 and Fig. 5, that the performance of the hybrid approach with *16* clusters is better than that with *4* clusters. Cluster size (or the number of nodes in a cluster) is one of the important parameters in determining the performance of our solution and is inversely proportional to the number of clusters. As the cluster size increases, the number of nodes that can provide certification service increases, but number of collisions also increases, that may result in loss of partial keys and certification packets. Smaller cluster size, reduces the initialization time, but is less tolerant to malicious nodes and have lower availability. When the number of clusters is *16* (i.e. the number of nodes in a cluster is small) the performance is better than when it is *4*, because, as the number of nodes in the cluster increases, collisions increase, and many of the partial key distribution packets and signed certificates are dropped. We will investigate the effects of cluster size in more detail, to determine the optimal size for the most efficient solution.

The minimum cluster size can be determined by the number of tolerable malicious nodes $k-1$ in a cluster. For the solution to be reliable and available, the number of nodes signing the certificate should at least be k and thus the total number of nodes in a cluster (including the node requesting for the certification service) (CS) is governed by the inequality,

$$CS \geq 2k+1$$

The maximum cluster size is related to the storage capacity of the nodes. In this algorithm, each node stores information about every other node in the cluster, and hence if the maximum storage capacity of a node is M,

$$CS \leq M+1$$

B.3 Delay characteristics

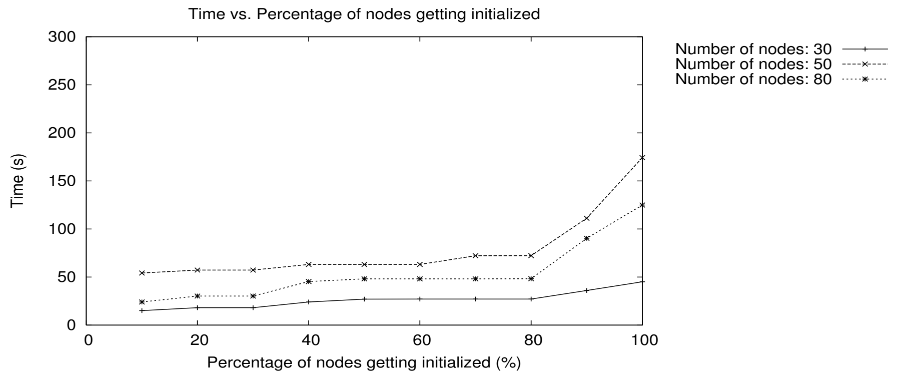

Fig. 6. Comparing initialization time for varying number of nodes

Fig. 6 shows the performance of the hybrid scheme with respect to initialization time for varying network sizes. The maximum node speed is set to *5 m/s*, and the number of clusters is *4*. The initialization time is the time for contributory key generation and distribution of the partial keys. The figure shows that the algorithm scales well for large networks and the initialization process is completed within a reasonable time period. For this particular scenario, *90 %* of the nodes in the network are initialized within the first *100 seconds*, while all the nodes are initialized within the first *150 seconds*. We observe that, even when the number of nodes in the cluster is large (i.e when the number of clusters is *4*), the initialization time is still small.

The delay in obtaining the first certificate represents the time taken for a new node in the network to start secure communication. This is an important parameter because it shows the delay introduced due to the hybrid protocol, and also gives the time at which nodes can start communicating securely. Fig. 7 shows the graph representing the average certificate generation delay for different network size and for two different mobility settings. The number of clusters is *4*. We observe that the average delay for obtaining certificates is less than *20 seconds* and the solution scales well to large networks and is robust with respect to mobility. It can be observed that average certification delay decreases with increase in network size. This is because of increased availability for providing certificate services that offsets the increased delay due to key generation and distribution.

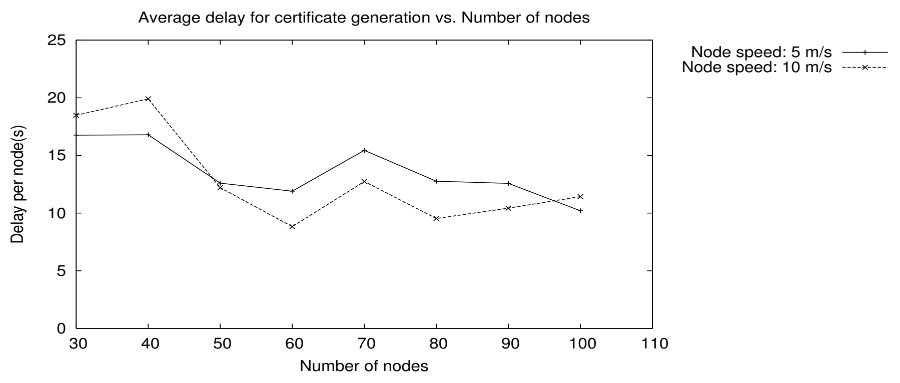

Fig. 7. Comparing certificate generation delay for different node speeds

VI. Conclusions and Future work

In this paper, we have presented a key management solution for use in ad hoc wireless networks. Our approach combines principles from both asymmetric and symmetric key cryptology to develop a novel hybrid solution for key generation and distribution. The algorithm provides a decentralized key exchange framework for symmetric encryption and a distributed certification authority used for asymmetric systems. The proposed methodology overcomes the limitation of purely symmetric or purely asymmetric systems and is designed to be scalable and robust. Analysis and simulation clearly show the advantages of the solution, particularly in terms of improved scalability, security and performance.

Current research efforts are towards determination of the optimal cluster size and the most suitable clustering algorithm for our solution. Extensive evaluations will be performed to compare our solution with other symmetric, asymmetric and cluster-based solutions. We are also investigating the applicability of this approach for large scale sensor networks.

References

[1] J-P.Hubaux, "What could we submit next year to WiSe," in *Invited talk,Second Workshop on Wireless Security (WiSe)*, (San Diego, USA), September 2003.

[2] A.Shamir, "How to Share a Secret," *Communications of the ACM*, vol. 22(11), pp. 612–613, 1979.

[3] Y. Desmedt and Y. Frankel, "Threshold Cryptosystems," in *Advances in Cryptology, Crytpo '89* (G.Brassard, ed.), (Santa Barbara, California), pp. 307–315, Springer-Verlag, August 1990.

[4] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," *IEEE Network*, vol. 13, no. 6, pp. 24–30, 1999.

[5] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing robust and ubiquitous security support for mobile ad hoc networks," in *Proceedings of the Ninth International Conference on Network Protocols (ICNP'01)*, p. 251, IEEE Computer Society, 2001.

[6] M. Bechler, H.-J. Hof, D. Kraft, F. Pahlke, and L. Wolf, "A Cluster-Based Security Architecture for Ad Hoc Networks," in *Proceedings of the 23rd IEEE Infocom*, March 2004.

[7] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of the 10th ACM conference on Computer and Communication Security*, pp. 52–61, ACM Press, 2003.

[8] S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks," in *Proceedings of the 10th ACM conference on Computer and communication security*, pp. 62–72, ACM Press, 2003.

[9] A. C-F.Chan and E. S. R. Sr., "Distributed Symmetric Key Management for Mobile Ad hoc Networks," in *Proceedings of the 23rd IEEE Infocom*, March 2004.

[10] S. Yi and R. Kravets, "MOCA: Mobile Certificate Authority for Wireless Ad Hoc Networks," in *Annual PKI Research Workshop Program*, (Maryland), April 2003.

[11] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of CRYPTO 84 on Advances in cryptology*, pp. 47–53, Springer-Verlag New York, Inc., 1985.

[12] W. A. A. Aram Khalili, Jonathan Katz, "Toward Secure Key Distribution in Truly Ad-Hoc Networks," in *2003 Symposium on Applications and the Internet Workshops*, IEEE Computer Society Press, Jan 2003.

[13] J.-P. Hubaux, L. Buttyan, and S. Capkun, "The Quest for Security in Mobile Ad hoc Networks," in *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pp. 146–155, ACM Press, 2001.

[14] P.Zimmermann, "The Official PGP User's Guide," 1955.

[15] N. Asokan and P. Ginzboorg, "Key-Agreement in Ad-hoc Networks," *Computer Communications*, vol. 23, no. 17, pp. 1627–1637, 2000.

[16] M. Steiner, G. Tsudik, and M. Waidner, "CLIQUES: A New Approach to Group Key Agreement," in *Proceedings of the 18th International Conference on Distributed Computing Systems ICDCS'98*, (Amsterdam), pp. 380–387, IEEE Computer Society Press, 1998.

[17] Y. Burmester. M., Desmedt, "A Secure and Efficient Conference Key Distribution System," in *Advances in Cryptology - EUROCRYPT'94*, (Berlin), pp. 275–286, Springer, 1994.

[18] R. S. L. Lamport and M.Peace, "The Byzantine Generals Problems," in *ACM Transaction on Programming Languages*, pp. 382–401, July 1982.

[19] Z.J.Haas and M.R.Pearlman, "A Zone Routing Protocol for Ad hoc Networks," in *Internet Draft, Internet Engineering Task Force*, November 1997.

[20] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "SPAN: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," in *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking*, (Rome, Italy), pp. 85–96, July 2001.

[21] S. Basagni, "Distributed Clustering for Ad Hoc Networks," in *Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '99)*, p. 310, IEEE Computer Society, 1999.

[22] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-hellman key distribution extended to group communication," in *Proceedings of the 3rd ACM conference on Computer and communications security*, pp. 31–37, ACM Press, 1996.

[23] G. Ateniese, M. Steiner, and G. Tsudik, "Authenticated group key agreement and friends," in *Proceedings of the 5th ACM conference on Computer and communications security*, pp. 17–26, ACM Press, 1998.

[24] W. Stallings, *Cryptography and Network Security.* Prentice Hall, 1999.

[25] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," *Communications of the Association for Computing Machinery*, vol. 21, pp. 120–126, Feb. 1978.

[26] F. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks," in *Security Protocols, 7th International Workshop Proceedings*, pp. 172–194, 1999.

[27] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive secret sharing or: How to cope with perpetual leakage," 1995.

[28] M. Stadler, "Publicly verifiable secret sharing," in *Advances in Cryptology — EUROCRYPT '96* (U. Maurer, ed.), vol. 1070 of *Lecture Notes in Computer Science*, pp. 190–199, Springer-Verlag, 1996.

[29] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad hoc Networks," in *Mobile Computing and Networking*, pp. 255–265, 2000.

[30] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks," in *Workshop on Parallel and Distributed Simulation*, pp. 154–161, 1998.