Providing Distributed Certificate Authority Service in Mobile Ad Hoc Networks

Y. Dong[1], H. W. Go[2], A. F. Sui[2], Victor O. K. Li[1], Lucas C. K. Hui[2], S. M. Yiu[2]

[1]Department of Electrical and Electronic Engineering
The University of Hong Kong, Hong Kong, China
{ydong, vli}@eee.hku.hk

[2]Department of Computer Science
The University of Hong Kong, Hong Kong, China
{hwgo, afsui, hui, smyiu}@cs.hku.hk

Abstract — **In this paper, we propose an architecture for providing distributed Certificate Authority (CA) service in Mobile Ad Hoc Networks (MANET), based on threshold cryptography. We have two major contributions: 1) we make use of the cluster structure to provide CA service, and design a scheme for locating CA server nodes in MANET; 2) we provide a proactive secret share update protocol, which periodically updates CA secret shares with low system overhead. Compared with existing approaches, our CA architecture provides faster CA services to user nodes at reduced system overhead.**

Index Terms —**Mobile ad hoc network, PKI, threshold secret sharing, cluster-based architecture, proactive update, distributed CA services.**

1. INTRODUCTION

A MOBILE Ad Hoc Network (MANET) is a self-organized wireless network without infrastructure. With the increasing popularity of MANET, its security issue has drawn greater attention.

Public Key Infrastructure (PKI) [8] is a widely used security mechanism in communication networks. In PKI, each entity has a public and private key pair. There is a trusted-by-all centralized authority, called Certificate Authority (CA), for key management. The CA has a public and private key pair, and signs public key certificate (PKC) for each user's public key using its private key. The PKC signed by the CA can then be used as a validity proof for the public key of the user. The CA is also responsible for PKC update or renewal, and revocation.

PKI has been considered as the foundation of security services in MANET, since it is efficient in key management and distribution, and is convenient to achieve authentication and no-repudiation. However, there are challenges to implement PKI in MANET. The conventional single-CA architecture in PKI will obviously suffer the single point of failure problem. Further, in the mobile environment of MANET, how to locate the CA is a non-trial issue, which may involve much communication overhead.

In [1], a decentralized CA method is proposed to solve the single point of failure problem, which distributes the functionality of a single CA to a set of nodes by secret sharing and threshold cryptography. Then any CA service has to be performed jointly by t CA nodes, where t is called the threshold of the secret sharing. In this way, even if an attacker has discovered the secret shares of more than one but less than t CA nodes, the attacker still cannot recover CA's secret key. However, the above threshold secret sharing scheme still fails when the shares of more than t CA nodes have been discovered by the intruders over a sufficiently long period. To enhance security, schemes using proactive share update [2], [9]-[11] has been proposed [1], in which a new set of shares are computed after a certain time interval. Therefore, an attacker has to complete the attack within this interval; otherwise all efforts are in vain after the shares change.

There are still unsolved issues with this distributed CA and proactive update scheme. First, the CA locating problem becomes aggravated, since a user node has to find t CA server nodes, much more difficult than finding one. Schemes such as flooding [7] are not viable since it will consume too much network resource. Second, efficient update of the secret shares in all CA nodes is not trivial. Although a scalable update scheme has been proposed in [5], it assumes every node is a CA and each node has at least t neighbors, which are unrealistic in MANET.

Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'0
0-7695-2369-2/05 $20.00 © 2005 IEEE

Although the CA location problem is solved by this approach, attackers are also alleviated with the concerns on which node to attack: they can attack any more than t nodes since every node is a CA.

In this paper, we have two contributions in resolving the above issues. First, we propose a clustering architecture for the organization of MANET, in which nodes are grouped into clusters. Each cluster has a cluster head (CH), through which CA server can be quickly located. Second, we devise a distributed secret share update scheme utilizing the sequential share update scheme proposed in [5]. By utilizing the information at CHs, secret shares can be quickly updated. From analysis and simulation, we show that our approach is able to achieve good service availability and quick CA response time, at reasonable communication overhead. Further, since the CAs in our scheme is confined to a subset of the total node population, the system security is enhanced compared to the case when all nodes have a secrete share [5].

The rest of the paper is organized as follows. In Section 2, we describe the system architecture. Section 3 presents the PKC scheme. Section 4 shows the details of the CA share update process. Analysis and discussions are presented in Section 5. Simulation results are provided in Section 6. Conclusion is given in Section 7.

2. System Architecture

We consider a mobile ad hoc network with N nodes, structured into clusters. In this paper, we do not specify any rules for cluster formation, since it is a separate issue and has been treated in work such as [6]. At the inception of the network, we assume that n CA nodes have been selected.

We select a cluster head (CH) for each cluster, based on the following rules: 1) when there is at least one CA node in the cluster, the CH has to be a CA; 2) when there is more than one CA in the cluster, the CA node with the smallest ID is chosen as CH; 3) when there is no CA in the cluster, an arbitrary node can be CH. Specifically, for a CH satisfying rule 3), the CH is only delegated to managing and distributing CA information, but will not participate in the CA share update stage. Therefore, in later discussions on share update, the participating CHs have to be selected based on rules 1) and 2).

When a user first joins the network, we assume it has been authenticated [4]. With this authentication, the user will approach the CA server nodes for issuing a digital PKC.

We assume an attacker can only obtain the secret share possessed by a CA node by observing the signatures signed by the CA. However, the attacker *cannot* hack into any CA node. Thus, if the secret share is changed, the attacker *will not* automatically obtain the new share. We also assume the information exchange between CAs is secured by the CA signatures.

3. Scheme for Providing PKC Service

In this paper, we consider the problem of how a user can obtain the PKC after joining the network. We adopt the (t,n) threshold scheme for the issuing and renewal of the PKC. In such a scheme, t out of n CA nodes are selected for the joint signing of the PKC. Therefore, for a user to obtain a PKC, it is equivalent to finding at least t CA servers for the certificate service.

A. CH-Assisted CA Locating Scheme

We devise a cluster-based scheme for locating CAs. The notations used in our scheme are summarized in Table 1.

t	Threshold
CH_i	cluster head of cluster i
U_i	user in cluster i
N_i	the number of CA nodes in cluster i
$B(i,j)$	the number of CA nodes in i's neighboring cluster j

Table 1: Notation of CH-assisted CA locating scheme

In our scheme, each CH maintains a CA information table (CIT), which contains a list of the CA nodes in its local cluster, and probably the CA information in other clusters. The procedure of locating CAs is as follows.

1. When U_i wants to locate at least t CAs, it sends a request to CH_i CA Information (CI).
2. CH_i collects CA distribution information, and passes it to U_i (to be explained below).
3. U_i selects t CA server nodes according to CI provided by CH_i, and sends renew request RENEW_REQ or new certificate request NEW_REQ messages to CAs. U_i either specifies the set of signer by passing the selected CA server

list to each one, or does not specify the set by using dynamic coalescing scheme proposed in [5].

4. CH_i combines the responding t partial signatures to generate the complete signature.

In Step 2, there are several possibilities when CH_i collects CI.

1) When $N_i \geq t$, there are enough CA in cluster i for CA service. Therefore, the CI sent to U_i contains the IDs of the CA nodes in the cluster.

2) When $N_i < t$, there are not enough CA in cluster i. In this case, CA in other clusters will have to be used. We require CH_i should periodically request its neighboring CHs to provide the number of CA in their own clusters. The interval of such a request is directly determined by the topology dynamics in a neighboring cluster. We further consider two sub-cases.

Case a) $N_i + \sum_j B(i,j) \geq t$, i.e., there are enough CAs in cluster i plus its direct neighboring clusters. Then CH_i records in the CIT both the CA ID in cluster i and $B(i,j)$. It is up to the user to contact the CH of a direct neighboring cluster for CA ID.

Case b) $N_i + \sum_j B(i,j) < t$, i.e., the number of CAs in both cluster i and its direct neighbors is not enough for the CA service. Upon receiving a user request, CH_i will send a request message to all other CHs, and each CH receiving the request message responds with a message indicating the number of CA in its cluster. This can be viewed as flooding within the CHs. It is obvious that the overhead is constrained when compared to the conventional flooding method.

Since the CA information could be piggybacked with the cluster management information (such as routing packets, cluster membership exchange, etc.), communication overhead can be correspondingly reduced, as discussed in Section 5.

4. CA Shared Secret Update

A. Scheme Overview

In our scheme, the CA services need to be performed by at least t server nodes. This alleviates the problem of single point of failure. To further enhance security, we require the shared secrets are updated periodically, as proposed in [5].

In this paper, we make use of the idea of sequential update proposed in [5]. However, this scheme is designed for share updates within one-hop neighborhood. For our cluster-based system, the scheme in [5] cannot be directly applied without modification. In this section, we propose a secret share update scheme for our cluster-based system. The process is composed of three phases: 1) update initialization; 2) update procedure; 3) update propagation. In the following we describe the details of the three phases.

B. Share Update Scheme

1) Phase One

We assume the MANET is coarsely synchronized among all the CAs. By "coarsely", we mean the CA clocks are within the accuracy of seconds (or even sub-seconds) relative to the "true" time. This assumption is reasonable in MANET [13]. All CAs maintain an update schedule, which specifies the time at which a share update should start.

At the inception of the update, at least t CAs should participate in the derivation of new secret shares, which is called new share initialization. It is after the initialization that the shares on other CAs can be updated based on this initial group of shares. Obviously, it is necessary to have one and only one initialization; otherwise the system will have multiple initial share groups, which will eventually lead to conflicts during the update process.

Thus, the goal of phase one (update initialization) is to ensure that there is indeed one initialization node group within the whole system. It is equivalent to finding one CA node, which will form a group with (no less than) t CA server nodes. To achieve the above design goal, we adopt the following measures.

A) We require that only a CA-CH node (a CA node that is also a CH) should be eligible as an initiator, in order to narrow the field of competition. To initialize the update, a CA-CH node will broadcast an initialization message to all other CA-CHs. This message contains the update request, and the ID of the sending CA-CH.

B) We adopt a random backoff scheme to further reduce the probability of multiple initializations. More specifically, a CA-CH will only broadcast the initialization message after a random backoff period.

The length of random backoff period is determined as follows. For a CA-CH node CH_i, it will choose in a uniform manner an integer W_i from the range $[0, CW_i]$, where CW_i is the contention window size. Then the node enters the backoff state, and waits for $W_i \times T_d$ seconds before broadcasting the

initialization message, where T_d is the time slot size common to all CA-CHs.

During the backoff period, CH_i will monitor the received messages. If it receives any initialization messages from other CA-CHs, the backoff state is aborted, and the node prepares for share update. After the completion of the backoff period, if the node receives no initialization messages, the node sends an initialization message, which is to be flooded among all CA-CHs. Since the flooding is only confined in CA-CHs, the involved overhead is constrained, even for large MANETs. Since different CA-CHs may choose different W_i values, it is likely that such a difference in backoff period will prevent many CA-CH nodes from transmitting initialization message.

The backoff period is determined by two parameters, T_d and CW_i. The selection of an appropriate T_d should be related to the largest propagation delay among CA-CHs within the MANET. For example, T_d can be chosen as the longest propagation delay between any two CA-CHs. In this way, any two CA-CHs node who choose W_i differing by 1 can avoid collision of initialization messages.

The selection of CW_i is dictated by the number of CA-CHs. A small CW_i increases the chance of collision, but a large CW_i prolongs the backoff time. An appropriate choice for CW_i is the number of CA-CH nodes. Since we require that at least t CA server nodes participate in the derivation of the new shares, it is desirable to prioritize the CA-CH nodes that have enough CA in its cluster, or at least in neighboring clusters. Generally, we can divide the contention window (CW) size into three categories. The first category, corresponding to CA-CH nodes having more than t CA nodes in the cluster, is assigned a small CW value, denoted as CW_1; for CA-CH nodes that have less than t CA nodes in the cluster, but more than t CA nodes plus the CA nodes in direct neighboring cluster, we assign a larger CW valued, denoted as CW_2; for the remaining CA-CH nodes (less than t CA nodes after counting direct neighboring clusters), we assign the largest CW value, CW_3. In this way, CA-CHs with more CA nodes in the vicinity have a higher chance of winning the competition.

C) Although the probability of collision can be lowered after the above two measures, it is still possible for multiple initializations to happen. Here we present the collision resolution method.

After a CA-CH node receives the initialization message, it sends an acknowledgement (ACK) to the sender. It is possible that a node (including a node that has sent an initialization message) will receive another initialization message afterwards. In this case, the node will compare the ID contained in the two messages. If the newer one has a smaller ID, the node will send an ACK to the new sender, and a negative acknowledgement (NACK) to the previous sender; if the newer one has a larger ID, the node will only transmit a NACK to the new sender. We require an initializing CA-CH node to collect ACKs from all CA-CHs before it can claim itself as the winner of the competition. Here we assume that each CA-CH has a complete list of CA-CHs in the network. Whenever there are multiple initializing nodes, the conflict will eventually be resolved, since only the initializing node with the smallest ID can be the sole winner.

2) Phase Two

After phase one, the winning CA-CH node, denoted as CA-CH_W, is to find (at least) t CA nodes to derive the new shares. This CA locating procedure is identical to the scheme described in Section 3, and will not be repeated here.

3) Phase Three

After phase two, at least t servers have updated their shares. Then the t updated servers will update the remaining CA nodes in the network. The goal of phase three is to propagate the share update to all the CA. When a CA finishes the share update process, it will inform its CH about update completion. The CH(s) collect update completion information from its local cluster, and inform the neighboring CHs that has not updated about the completion of the update. Then the informed CHs request the CA nodes in its cluster to update. The CAs will then contact the CHs to locate t CAs with new shares, which can be based on the locating method described previously.

5. Analysis and Discussions

In this section, we analyze the performance of the CA locating scheme and discuss the share update scheme.

A. Analysis on CH-assisted CA Locating Scheme

1) System Response Time

We define the system response time as the interval from the moment when a user starts seeking CA

service until the time when CA service is completed by the CA server nodes. When comparing the system response time of our scheme with other schemes, such as the flooding scheme in [7], the major difference in system response time lies in the time needed for a user to locate enough CA server nodes, which is called CA server locating delay (CSLD). We now analyze the CSLD in our scheme.

It is obvious that our CH-assisted CA locating scheme shifts the responsibility of CA discovery from each user node to CHs. According to our scheme, a user node only needs to contact the CH for the CA information table (CIT). As presented in Section 3, there are three possibilities on the CH reply, namely (see Section 3-B for the details): Case 1), when $N_i \geq t$; Case 2-a), when $N_i + \sum_j B(i,j) \geq t$; Case 2-b), when $N_i + \sum_j B(i,j) < t$. For all cases, after obtaining the CIT, the user's job is extremely simple: contact CAs in the cluster and/or CHs of other clusters as instructed by the CIT.

In Case 1), the user can immediately contact CA nodes within the cluster after receiving the CIT. Thus, the locating delay is just the response delay of the CH, and the system response time is mostly the time for CA server nodes to perform the CA service.

In Case 2-a), the user node needs to contact CA server nodes in neighboring clusters. Note that we have two approaches to maintain the CIT (see Section 3-B): the first one is to record the CA IDs of a neighbor cluster in the CIT, and the second one is to only record the number of CA nodes in that cluster. For the first approach, the delay is essentially the same as Case 1); for the second approach, the delay is a little, but not much, longer since the user node only has to communicate with several neighboring CHs to obtain the CA IDs.

In Case 2-b), since there are not enough CA in the vicinity of the requesting user, the CH has to collect the CA information from the remote area by broadcasting requests to all the CHs in the network. The CH can reply to the requesting user once the received responses consist of enough number of CA, instead of waiting for all the responses from all CHs. Thus, the locating delay is mainly determined by the response delay from other CHs, which should be quite limited.

When compared to the flooding scheme [7], the locating delay is almost negligible for Case 1) and Case 2-a) in our scheme; even for Case 2-b), the locating delay is still limited.

Above all, the system response time is greatly shortened, which is a salient advantage for our scheme.

2) System Overhead

In our scheme, a CH is required to maintain CA ID within the cluster. Since this work can be viewed as part of the job for monitoring the cluster membership, there is no system overhead from a CA service perspective. Therefore, we only have two sources of system overhead.

Firstly, the CH needs to periodically exchange CA information with the direct neighbors in both Case 2-a) and 2-b). However, in cluster management, information exchange (such as routing, cluster membership change, etc.) is frequent between neighboring CHs. Therefore, the CA information exchange can be piggybacked with other cluster management information. In this way, the extra system overhead is negligible.

Secondly, a CH needs to flood other CHs for CA information in Case 2-b) when receiving a user service request, during which a certain amount of overhead will occur. However, the overhead is much reduced when compared to other approaches. Further, we can also avoid the occurrence of such an undesirable situation by merging clusters with few CAs; by this means, most of CA services can be obtained within the vicinity. Another method is that the CH only chooses to probe a subset of CHs, e.g., the relatively closer CHs, for future report, after the initial flooding. This approach is better when the frequency of user CA service request is higher and the mobility of the nodes is slower.

Overall, our cluster-based approach is able to provide prompt system response while reducing system overhead to a minimum. Thus, our approach is particularly suitable for large MANET, which could be otherwise difficult to manage with the conventional methods.

B. Discussions on CA Share Update Scheme

Our share update scheme exhibits two desirable characteristics.

The first is that we guarantee there is only one initialization CA node within the whole MANET. This is achieved by: 1) random backoff to minimize the collision probability; 2) collision resolution by prioritizing the node with the smallest ID. The share update scheme presented in [5] does not consider such an issue, and will thus suffer from the problem of multiple initializations.

The second feature of our scheme is that we utilize CH-assisted CA locating scheme for CA share update and propagation. Such an approach simplifies the implementation complexity, since functions for CA locating can be re-used. Further, our scheme provides an orderly update procedure, as it propagates from one cluster to another. Correspondingly, the update process is also speeded up.

C. Discussions on Security

In [5], the secret shares are distributed to all nodes in the network, instead of to a subset of selected nodes as in paper [1]. The problem of locating CA nodes is solved, since a node requesting CA service only needs to find more than t neighboring nodes. However, this approach greatly increases the number of shares in the network. Clearly, the more number of shares, the higher is the probability for attackers to compromise more than t shares.

Let p be the probability of one node being compromised, p_f be the probability of a CA secret being compromised in the fully distributed CA model in [5], and N be the total number of nodes in the network. Then

$$p_f = p\{K \geq t, K \in (1, N)\} = \sum_{i=t}^{N} C_N^i p^i (1-p)^{N-i} \qquad (1)$$

Given t, since p_f is monotone increasing with N, it is shown in Equation (1) that a larger system is more prone to attacks. Therefore, the approach in [5] is not desirable for large networks.

In our work, the number of CAs is much smaller, and thus it will have better security than [5]. Let p_p be the probability of the secret being compromised in our partially distributed CA model. There is

$$p_p = p^{'}\{K \geq t, K \in (1, n)\} = \sum_{i=t}^{n} C_n^i p^i (1-p)^{n-i} \qquad (2)$$

Obviously, p_p will be much smaller than p_p since $n \ll N$.

D. Comparison

In this section, we compare our work with the other related work. To provide CA services, there are two extreme cases. One is the traditional centralized CA, and the other is the fully distributed case [5] in which every node in the network is a CA. Our method can be categorized as partially distributed in the sense that only some of the nodes hold the shares. In following Table 2, we summarize the performance comparison of these three strategies.

Schemes / Metrics	Centralized CA	Partially distributed CA	Fully distributed CA
Communication overhead	**high**	**medium**	low
Service response time	**long**	**quick**	quick
CA service Availability	**low**	**high**	high
Number of share-holding nodes	**1**	**n**	N (much larger than n)
Single point of failure	**yes**	**no**	no
Ease to locate enough CA nodes for attack	difficult	not easy	easy

Table 2: Comparison of the centralized, partially and fully distributed CA

6. SIMULATION RESULTS

A. Simulation Setup

We use NS2 as the simulation tool. IEEE 802.11 MAC is adopted, and the radio transmission range is 250 m. We simulate an ad hoc network with 60 nodes, among which 20 are CA nodes. The nodes are uniformly distributed within an area of 600m by 600m. The node movement follows the widely used mobility model implemented in the CMU Monarch extension. We simulate two maximal moving speeds, 1 m/s and 5 m/s, and the nodes have zero pause time.

We implement the distributed clustering scheme in [6] to form one-hop non-overlapping clusters. Since the cluster head selection rule in [6] is based on the smallest ID, we modify the selection rule so that the cluster head is a CA whenever possible.

In the simulation, we investigate the probability of locating enough CAs in a cluster and its neighbors. It can be expected that when a cluster contains enough CA nodes, the CA locating time is almost negligible, since the node information is contained in the response from the CH node. When the CAs in the directed neighbors need to be involved, the system response time is still short, because the requesting node only need to contact the CHs in the neighboring clusters. Therefore, we focus on investigating the CA distribution within one cluster and its neighbors.

B. CA Distribution within One Cluster

We first investigate the per node probability of the number of CAs within a cluster. In the simulation, we track a particular node and count the number of CAs

within its cluster at a snapshot of the network, based on which the per node probability is calculated.

In Figure 1, we plot the per node probability mass functions (pmf) of the number of CA nodes within a cluster when nodes are traveling at speeds of 1 m/s and 5 m/s. It should be noted that we use curves in the following figures only for the purpose of illustration. In fact, the number of CAs is a discrete value. As we can observe, the probability mass functions are similar for different mobile speeds, and this is expected. Since nodes are moving randomly, the node distribution (including CA nodes) is still uniform and is not affected by the mobile speed.

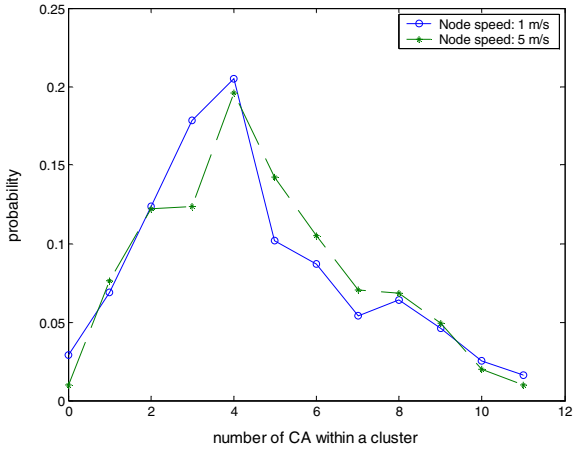

Figure 1: Probability mass functions of the number of CAs in a cluster.

In Figure 2, we show the probability that the number of CA in a cluster is greater than or equal to the CA threshold t. Obviously, when t is a small number such as 2 or 3, there is a high probability that the CA service can be performed within the cluster. Even when t is 4, the probability is still as high as 0.5.

On the other hand, the probability decays fast when t is greater than 4. Therefore, it is necessary to balance between t and the number of CAs. From our experiment, we find there are on the average 4 CAs within the cluster for a particular node, and the probability for t no greater than 4 is about 0.5. Therefore, as a rule of thumb, we may select t to be smaller than the average number of CAs in a cluster, and the chance of finding enough CAs within the cluster should be higher than 0.5. Of course, the selection of t also depends on the desired security level. Obviously, the selection of t is a tradeoff between service availability and security level.

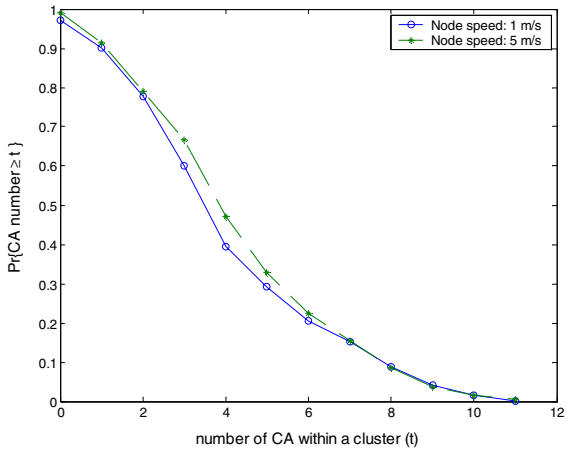

Figure 2: Probability of the number of CAs in a cluster greater than or equal to the CA threshold t.

C.CA in Neighboring Clusters

Here we study the per node probability of the number of CA in a cluster plus neighboring clusters.

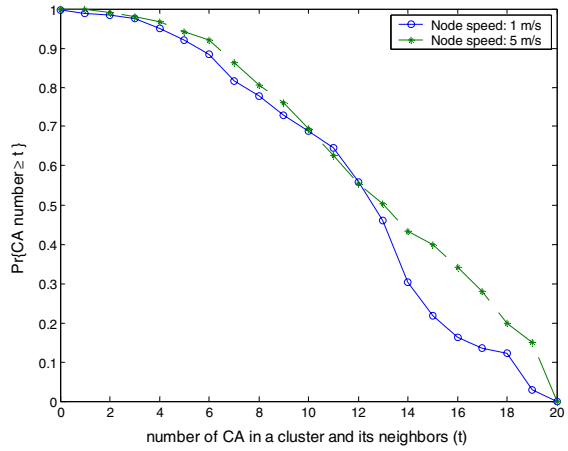

Figure 3: Probability of the number of CAs in a cluster and its neighbors greater than or equal to t.

In Figure 3, we present the probability that the total number of CAs in a cluster and its neighbors is greater than or equal to t, when node speeds are 1 m/s and 5 m/s. Again, there is slight difference between curves of the two speeds. This is because the number of clusters as well as node and CA distribution are not affected by the mobile speed.

Most importantly, we notice that the probability is high even for relatively large value of t. For example, when t is as high as 10, the probability for a cluster and its neighbors to have more than 10 CA nodes is still above 0.7. When t is moderate or small, the probability is even higher. For example, when t is 4,

the probability is above 0.95; when t is 6, the probability is still above 0.9. Therefore, for a moderate t, the probability for a search beyond neighboring clusters is rather low.

From the above simulation results, we observe that our protocol is able to achieve high availability of local CA service, i.e., within the same cluster and its direct neighbors. Obviously, this high availability of local service will translate into prompt system response time for CA services.

7. CONCLUSION

In this paper, we establish a cluster-based architecture to realize CA service in MANET. Our CH-assisted CA locating scheme shifts the responsibility of CA discovery from each user node to CHs. In this way, distributed CA information is managed only among the CHs, which greatly reduces service response time and system overhead. We also propose a share update procedure, which can resolve the multiple initializations problem and achieves fast system-wide update. Through simulation, we show that for a moderate CA threshold, there is a high probability to find enough CAs nodes within a cluster or in neighboring clusters. Consequently, system response time can be significantly improved. We also find node mobility has little influence on the availability of service.

ACKNOWLEDGMENTS

This research is supported in part by the Areas of Excellence Scheme established under the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-01/99), a grant from the Research Grants Council of the HKSAR, China (Project No. HKU/7144/03E), and a grant from the Innovation and Technology Commission of the HKSAR, China (Project No. ITS/170/01).

REFERENCES

[1] L. Zhou, Z. J. Haas, "Securing Ad Hoc Networks," *IEEE Network*, Nov/Dec 1999.

[2] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive Secret Sharing Or: How to Cope With Perpetual Leakage," *Advances in Cryptography – Crypto' 95*, Lecture Notes in Computer Science 963, 1998.

[3] A. Shamir, "How to Share a Secret," *Communications of the ACM*, 22(11): 612-613, Nov. 1979.

[4] M. Bechler, H,-J. Hof, D. Kraft, F. Pahlke, and L,Wolf, "A Cluster-Based Security Architecture for Ad Hoc Networks," *Proceedings of the 23rd IEEE INFOCOM'04*, Hong Kong, China, Mar. 2004.

[5] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks," *Proceedings. Of the International Conference on Network Protocols (ICNP)*, pp. 251-260, 2001.

[6] C. R. Lin and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks," *IEEE JSAC*, Vol. 15, No. 7, Sep. 1997.

[7] S. Yi, and R. Kravets, "MOCA: Mobile Certificate Authority for Wireless Ad Hoc Networks," *2nd Annual PKI Research Workshop Program*, Apr. 2003.

[8] S. Kent, and T. Polk, IETF Public-Key Infrastructure Working Group Charter, Available at http://www.ietf.org/html.charters/pkix-charter.html.

[9] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive public-key and signature schemes," *Proceedings of the 4th Annual Conference on Computer Communications Security*, pp. 100–110, Zurich, Switzerland, Apr. 1997.

[10] Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung, "Proactive RSA," In *Advances in Cryptology—Crypto'97,* Lecture Notes in Computer Science 1294, pp. 440–454. 1997.

[11] Y. Frankel, P. Gemmel, P. MacKenzie, and M. Yung, "Optimal resilience proactive public-key cryptosystems," *IEEE Proceedings of the 38th Symposium on Foundations of Computer Science*, pp. 384–393, Oct. 1997.

[12] D. Balfanz, D. K. Smetters, P. Stewart and H. Chi Wong "Talking To Strangers: Authentication in Ad-Hoc Wireless Networks," In Symposium on Network and Distributed Systems Security (NDSS '02), San Diego, California, Feb. 2002

[13] T. Lai, D. Zhou, **"**Efficient, and scalable IEEE 802.11 ad-hoc-mode timing synchronization function**,"** *Advanced Information Networking and Applications*, pp. 318–323, Mar. 2003.