Summer 2005

Collaborative Caching for efficient and Robust Certificate Authority Services in Mobile Ad-Hoc Networks

Laith Abdulaziz Al-Sulaiman
Old Dominion University

COLLABORATIVE CACHING FOR EFFICIENT AND ROBUST CERTIFICATE AUTHORITY SERVICES IN MOBILE AD-HOC NETWORKS

by

Laith Abdulaziz Al-Sulaiman
M.S., December 2002, Old Dominion University
B.S., June 1999, King Saud University, Saudi Arabia

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
August 2005

Approved by:

Hussein Abdel-Wahab (Director)

Ravi Mukkamala (Member)

C. Michael Overstreet (Member)

Christian Wild (Member)

Min Song (Member)

ABSTRACT

Collaborative Caching for Efficient and Robust Certificate Authority Services in Mobile Ad-Hoc Networks

Laith A. Al-Sulaiman
Old Dominion University, 2005
Director: Dr. Hussein Abdel-Wahab

Security in Mobile Ad-Hoc Network (MANET) is getting a lot of attention due to its inherent vulnerability to a wide spectrum of attacks. Threats exist in every layer of MANET stack, and different solutions have been adapted for each security problem. Additionally, availability is an important criterion in most MANET solutions, but many security frameworks did not consider it. Public-Key Infrastructure (PKI) is no exception, and its deployment in MANET needs major design and implementation modifications that can fit constraints unique to this environment. Our focus in this dissertation is to adapt and increase the availability of Certificate Authority (CA) services, as a major PKI entity, in MANET.

Several attempts have been proposed to deal with the problem of deploying CA in MANET to provide a generic public-key framework, but each either ends up sacrificing system security or availability. Here, the main goal of our work is to provide a solution that addresses performance and security issues of providing MANET-based PKI. Particularly, we would like to maintain the availability of the services provided by CA while keeping the network's packet overhead as low as possible.

In this dissertation, we present a MANET-based framework suitable for exchanging public-key certificates by collaborative caching between MANET clients. We show that our system can meet the challenges of providing robust and secure CA services in MANET. Augmented by simulation results, we demonstrate quantitatively the feasibility of our work as we were able to reduce network overhead associated with

threshold based CA queries up to 92% as compared to related work in addition to having a very short response time. The dependency on CA servers has been reduced, and the system was able to tolerate as much as two-third inoperative CA servers without noticeable decrease in the service performance.

This thesis is dedicated to *Laila.*

ACKNOWLEDGMENTS

I praise God for all His bounties in my life, including this work.

This dissertation would not have been successfully completed without the contributions of a number of people. My deepest gratitude and appreciation are due to Dr. Hussein Abdel-Wahab for his valuable time, guidance, and encouragement throughout the program. I owe him much for the long hours of motivating discussions, constructive feedback, and careful review of this dissertation. Additionally, I would like to extend my thanks and gratitude to all my committee members: Dr. Ravi Mukkamala, Dr. C. Michael Overstreet, Dr. Chris Wild, and Dr. Min Song for their valuable feedback concerning this dissertation. Special thanks to Rex Hamaker for proofreading.

I will never forget my parents' prayers and thoughts, my brothers and sisters' support, and my friends' encouragements. My wife, Sara, deserves my warm thanks for her inspiration, sacrifices, and patience throughout this journey, and my daughter, Dana, for her lovely smile.

I would like to thank Imam Muhammad bin Saud University for their generous scholarship.

TABLE OF CONTENTS

Page

Section Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

SECTION 1
INTRODUCTION

Mobile Ad-Hoc Network (MANET) applications and services pose many interesting challenges due to the unique features of MANET environment. More specifically, MANET security is getting a lot of attention due to its inherent vulnerability to attacks. Threats exist in every layer of MANET stack, and different solutions have been adapted for each security problem. Another problem for MANET is availability, but adding more resources will not necessarily make the system more available. One important problem is how to provide a security framework that could be utilized in different MANET protocols and applications. One way of providing a generic, security-based framework is to deploy Public-Key Infrastructure (PKI). However, PKI deployment in MANET needs to be designed and tailored carefully to comply with MANET. More specifically, the focus of our discussion is how to adapt the Certificate Authority (CA), which is a major PKI entity, in MANET.

Several attempts have been made to tackle the problem of providing a generic public-key framework, but each either ends up sacrificing system security or availability. Here, the main goal of our work is to provide a solution that addresses performance and security issues of providing MANET-based PKI. Particularly, we would like to increase the availability of the services provided by CA component while lowering the packet overhead of the network without increasing network vulnerability.

In this dissertation, we present a framework suitable for exchanging public-key certificates in MANET based on collaborative caching between network clients. We will show that our system can meet the challenges of providing robust and secure CA services in such harsh environment. Supported by simulation results, we will also demonstrate quantitatively the feasibility of our work as compared to other related research which has

---

The journal model for this dissertation is the IEEE/ACM Transactions on Networking.

addressed the same problem. In this thesis, the terms framework, system, service, and protocol are used interchangeably. The term "Principal" will be used to refer to an entity involved in cryptographically protected communication (e.g. human or machine).

This section is organized as follows: Section 1.1 is a general overview of our work. Section 1.2 presents the motivation and objectives for our work while Section 1.3 is a summary of our results and contributions. Finally, Section 1.4 is the outline of this dissertation.

1.1. Overview

MANET is raising many interesting challenges for applications and services due to its unique characteristics. Not relying on fixed infrastructure combined with tight constraints, such as power consumption, transmission range, and possibly computational capabilities, are just a few examples of such challenges.

Security, on the other hand, is as crucial as other challenges and has received much attention from researchers in every aspect of MANET due to its inherent vulnerability to wider security attacks compared to wired networks. Assuming that nodes will be safe from physical attacks is no longer a realistic assumption, not to mention other diverse threats such as spectrum jamming, disruption of routing messages, denial of service, and eavesdropping (e.g.[41], [73], [57], [28]).

Availability is another significant problem, and as an important dimension of Quality of Service (QoS), has been addressed in MANET from a variety of perspectives (e.g. [14], [27], [45]). Unfortunately, availability cannot be guaranteed by just adding more resources without careful planning. Naively adding more resources may have adverse effect like increasing the overhead of the system or exposing its security.

MANET requires solutions that consider its unique characteristics. Thus, solutions designed for wired networks cannot be ported transparently because, simply, many of them have been proposed under the assumption of good connectivity and no mobility, which means a low link failure between nodes.

Another problem in MANET is adapting security frameworks to protect various components of MANET stack. Various public-key based solutions such as [5], [30] seem to be good candidates, but they are too expensive to be practically deployed unless we redesign them using a MANET-centric approach. A typical PKI system consists of certificates, a CA, a revocation mechanism, and a mechanism to evaluate a chain of certificates to the target [41]. Public/private-keys can be used for encrypting and/or signing in many vital modules and applications, such as authentication, secure exchange of routing information, encrypting or signing emails, and much more. A CA is usually used to organize, store, and issue certificates that associate a public-key to an identity.

As we can see, PKI has been used and implemented in wired networks and, in principle, could be used to provide security for MANET as well. However, implementing PKI in MANET is not an easy task since current PKI solutions are conceptually designed for centralized, wired, and well-connected networks. The unique features of MANET makes the task of providing a reliable and secure service much more difficult.

A direct solution like installing a single CA server per MANET will make it very vulnerable and, in fact, will create a single point of failure. Moreover, replication of the CA could increase robustness but would make the situation even worse, compared to the single CA solution, by exposing the system and giving the attacker more choices.

Recently, researchers have tried to solve the dilemma of security and availability and provided some solutions to adapt CA services to MANET. These solutions rely on secret sharing concepts and mechanisms (e.g. [25], [39], [66], [26]) to increase system security and availability. Conceptually, secret sharing is a mechanism of distributing a secret (e.g. key) between multiple, say n, parties in a way such that any t (also called "threshold" where $1 \leq t \leq n$) parties can reveal the secret. This means that if less than t parties have been compromised, the secret is still safe. When $t = 1$, it means the system is more available since any one out of n could be contacted, and when $t = n$, it means the system is highly secure since an attacker needs to compromise all the nodes to gain the secret. However, the system may lose its robustness since compromising a single node will put the system out of service. Thus, t is the balance point between availability and

security. On the bright side, secret sharing could be a natural fit for MANET; however, it cannot be deployed without consequences, as we will see later in this dissertation.

The first attempt to adapt CA service based on secret sharing techniques to MANET was by Hass and Zhou [73]. They sketched an elegant solution based on Cornell Online Certificate Authority (COCA) as a foundation for a pervasive CA service in MANET. The novelty was to use threshold cryptography to securely distribute the private-key of the service over multiple servers. Each server has part of the CA service private-key, and it must collaborate with at least $t-1$ other servers to provide security services such as signing and/or updating certificates. When a client wants to use the service it will contact a predetermined CA server called the "delegate." The delegate will contact $t-1$ CAs passing the request to them and wait for their partial signature for the certificate in question. After collecting the replies, the delegate will reconstruct the certificate and send it back to the client.

Inspired by COCA, Mobile CA (MOCA) [70] has been introduced as a candidate for providing CA services in MANET but with more consideration to its unique properties since COCA was originally designed for wired networks and did not consider client connectivity. MOCA consists of mobile nodes within a MANET selected to provide distributed CA functionality. MOCA nodes apply threshold cryptography to share the responsibility and increase the availability of the service. Client nodes need to contact sufficient MOCAs to get the service. When a client wishes to get a certificate, it sends request packets to at least t MOCA servers and waits for a reply. Any MOCA that receives a request will answer by sending a reply that contains its *partial* signature for the subject in question. The client needs to receive at least t valid replies and combine them to reconstruct the whole certificate. Usually, MOCA clients will send more than t unicast requests to increase protocol robustness.

Another proposal, Ubiquitous and Robust Security Architecture (URSA), was suggested by Kong *et al.* [43], and their goal was to provide pervasive CA services by making all n nodes in the network share CA functionality. However, since sharing the private-key by *all* participants is obviously unfeasible, especially in MANET, Kong's

scheme was to distribute the service private-key into every coalition of nodes, where each coalition has at least t nodes, and they are one hop apart such that a client needs only to contact t instead of n nodes by sending a local broadcast request to obtain service. Once a certificate is issued, it is trusted by the whole network since it has been signed by the same private-key. Thus, a well-defined issuing policy is needed to control certificate issuing.

Hubaux el al.[36] suggested a scheme similar to Pretty Good Privacy (PGP) scheme [41] in which there is no concept of centralized CA and each node acts as its own CA. Every node will build its own "Web of Trust;" thus, there is no longer a well known trusted third party, as in the case of conventional PKI.

1.2. Motivation and Objectives

In the previous section, we showed various approaches that consider solving key management problems associated with MANET. We think that they all share the shortcoming of favoring security against availability or vise versa.

Zhou and Haas [73] introduced a novel concept of providing a robust CA service by disseminating the key over several servers using threshold cryptographic techniques. Although their work addressed security and fault tolerance issues, the scheme is conceptual, and the connectivity between clients and CA servers has not been addressed. Yi and Kravets [70] extended COCA concepts and have provided detailed performance evaluation to their work and compare it to their earlier prototype which is based on pure flooding [71]. The performance criteria for evaluating MOCA was to measure the ratio of certificate requests and reply messages over the total messages obtained when pure flooding is used. The MOCA scheme was 5-30% more efficient than pure flooding in terms of packet overhead. On the other hand, the URSA [43] scheme has solved the problem of availability by allowing any t nodes to sign certificates, assuming that at least t one-hop neighbors are available. As a consequence, they made t small to guarantee availability, system security is sacrificed. The Hubaux [36] solution is suitable for small self-organized networks in which every node builds its own trust relations, and it has the

shortcoming of scalability and the lack of a definite trust anchor (i.e. CA) which is trusted automatically by definition. Moreover, the framework does not use threshold cryptography, but we mention it to complete our coverage of the subject. We are going to elaborate more on the abovementioned work in Section 3.

We believe that there is room for improvement, and such enhancements could be accomplished by trying to make CA services have better performance while enjoying similar security. Here, we are presenting our work, CACMAN, which stands for "CAching Certificates in Mobile Ad-hoc Networks;" and the objective of this work could be stated as follows:

> *The aim of CACMAN is to minimize the burden of adapting CA services in MANET and reduce packet overhead while maintaining high availability of the service without compromising its security.*

This goal has been reached by allowing clients to share some responsibilities with CA servers by cooperatively caching portions of the certificates generated by CA servers. The characteristics of certificates, such as relatively long validity periods and modest storage requirements, can make caching a reasonable solution to reduce the burden of offering highly available and secure services by relieving CA servers from some of their duties. Our cashing-based framework has addressed the security and performance challenges of providing CA services in MANET.

1.3. Methodology, Result, and Contribution

CACMAN can be viewed as a hybrid between URSA and MOCA that boasts the best of both worlds, i.e. the availability of the former and the security of the latter. Such fusion, supported with novel techniques, shows CACMAN's strength. We chose adhering to secret sharing principles to reach our objective, and we have provided quantitative measurements showing how CACMAN contribute to solving the aforementioned problems. Network simulator NS-2 [47] is been used to implement CACMAN and enabled us to evaluate the significant of various parameters in relation to our objectives.

CACMAN have two major operations: Certificate Request (CREQ) and Certificate Reply (CREP). Any client that wants to obtain another principal's certificate will search its own cache to see if it has this certificate; otherwise, it will identify the missing partial certificates and proceed by sending CREQ by local broadcast. Every CACMAN client that receives a CREQ message inspects its cache for a possible hit, and a CREP is sent back to the requester if a full or partial certificate(s) is found for the subject. After that, the client waits for a specified time; if it receives sufficient responses, then it reconstructs the certificate. If the requester times out because it has not received sufficient partials, or the reconstruction step has failed, then it may try to contact CA servers by flooding the network or by sending unicast messages. Otherwise, it reports a failure to the client or retries later.

The novelty of CACMAN is based on the concept of applying caching techniques to boost the performance of CA service. We proposed and implemented several techniques starting with "CACMAN-basic" model and extending it to "CACMAN X-Replies," "CACMAN Lazy-Replies with Cancellation," and "Cache-Focus" variants. Those techniques and enhancements vary in their requirements, assumptions, overhead, and tradeoffs. Every proposed technique is supported by NS-2 simulation results. We also show the consequences of implementing revocation schemes in our work and discuss different design and deployment issues.

We can summarize our contributions in this dissertation into the following main points:

1) *Reducing Network Overhead*: CACMAN is able to achieve a significant reduction of packet overhead which varies with every configuration between 74% and 91% as compared to other related protocols.

2) *Reducing Servers' Overhead*: Simulation results show significant decrease on CA servers' load which enabled us to achieve the following desired goals:

    a. *Fewer Requests per CA*: This was due to clients' contribution in which they took part of CA responsibilities and volunteered portions of their

memory to cache certificates. CA load has been reduced dramatically compared to MOCA and flooding.

b. *Allowing more Offline Time*: In CACMAN, a CA server is no longer required to be online to provide a continuous service and it can go offline for maintenance or to reduce its physical exposure to threats.

3) *Separating Availability from Security:* the value of t in a threshold-based secret-sharing system controls the two aspects of the system which we want to keep as high as possible. Unfortunately, they are inversely proportional (i.e. availability vs. security). In CACMAN, we are able to make balance availability and security.

4) *Maintaining High Availability*: CACMAN allows service to be up most of the time, even if most of the CA servers are destroyed (we simulated the scenario where 20 out of 30 CA were damaged) without noticeable degradation of performance.

5) *Reducing Response Time*: Our framework is able to achieve higher response time, up to 16 times faster, than other related frameworks based on our simulation results.

6) *Obeying Security Requirements*: In CACMAN, we obeyed most of the security assumptions of related work, such as threshold concepts.

Table I shows how CACMAN can maintain a high success ratio for certificate requests with much lower packet overhead as compared to MOCA (reported in the first row). Notice that while success rates are comparable to MOCA, network traffic overhead is reduced.

TABLE I
SAMPLE VALUES FOR CACMAN COMPARED TO RELATED WORK ($t = 15$)

Model	*Ratio to flooding*	*Success Ratio*
MOCA Closest unicast	86.2%	77%
CACMAN basic,150 cache size	9.7%	63.3%
CACMAN all-reps,150 cache size	14.7%	78.9%
CACMAN basic, 225 cache size	11.1%	73.9%
CACMAN all-reps, 225 cache size	20%	89%

1.4. Dissertation Outline

The organization of the dissertation is as follows: In Section 2, the necessary background that covers MANET security essentials and basics is given. It starts by discussing Public/Private-key cryptography and how it differs from conventional methods. The PKI that governs key deployment and management is also discussed with some focus on validation and revoking schemes. A mathematical overview of how to share a key between multiple parties is presented with more description on how to make it more practical and robust. The section also introduces some characteristics and challenges of MANET followed by a closer look at routing challenges. Then, the focus is on various MANET specific security challenges, like availability, routing, intrusion detection, and key management. Section 3 gives more attention to various key management schemes and discusses related work (e.g. URSA, COCA, and MOCA). Section 4 is devoted to our proposed solution, CACMAN, from a design perspective . It begins with a detailed description of our framework followed by its assumptions. Protocol description is given followed by a discussion of alternative ways of communication. Various enhancements to CACMAN are presented, followed by a discussion of the impact of applying different revocation schemes. The section continues with a discussion of other miscellaneous issues that affect the implementation of the framework, followed by sketches of practical deployment scenarios. Section 5 compares CACMAN performance to related work. It begins by describing the simulation environment and how we conducted the experiments followed by simulation assumptions. The results of simulating various CACMAN enhancements and other configurations are visualized and tabulated accompanied by a

detailed analysis. Section 6 contains the conclusion and possibilities for future work. This thesis also contains appendices that show further details and other aspects of our work. Appendix A contains an overview of the simulation environment used to implement CACMAN. Appendix B contains the implementation information for CACMAN. Although it can be presented as a separate section, we preferred to move the design and implementation details to appendices as it does not affect the basic understanding of the work.

SECTION 2
BACKGROUND

This section is a presentation of some preliminary background. It will cover three general areas: security related background, MANET overview, their fusion, MANET security.

Security in MANET has received a lot of attention and the problem is clearly so broad that it is difficult to reach a single general solution. Thus, each proposal addresses the issue from a different perspective accompanied by different assumptions. Solutions are diverse as they focus on various aspects of MANET stack targeting application, end-to-end, routing, or MAC level security problems.

The organization of this section will be as follows: Section 2.1 is a cryptography introduction. Section 2.2 is a general overview of Public-key Infrastructure (PKI). In Section 2.3, we will focus on certificate revocation mechanisms. Section 2.4 is a brief background of secret sharing. Section 2.5 is dedicated to showing how to use secret sharing practically to do cryptographic operations. Section 2.6 is a general overview of MANET. Section 2.7 will cover general security problems faced in MANET, and Section 2.8 is a summary.

2.1. Public-Key Cryptography

Cryptographic functions are categorized into three types: secret key functions, public-key functions, and hash functions [41]. The difference among these functions is represented in Fig. 1. In this section, we will refer to pubic/private-key cryptography more often than other schemes; thus, we will cover a few more details about it. Usually, each principal has two pairs of keys, i.e. public and private. The public-key, as the name reflects, is known by others, and the private-key is known only to its owner. When someone wants to send an encrypted message so that no one other than the receiver can view it, the sender will use the public-key of the receiver to encrypt the message. For the principal to decrypt the message, it should use its private-key to decrypt it.

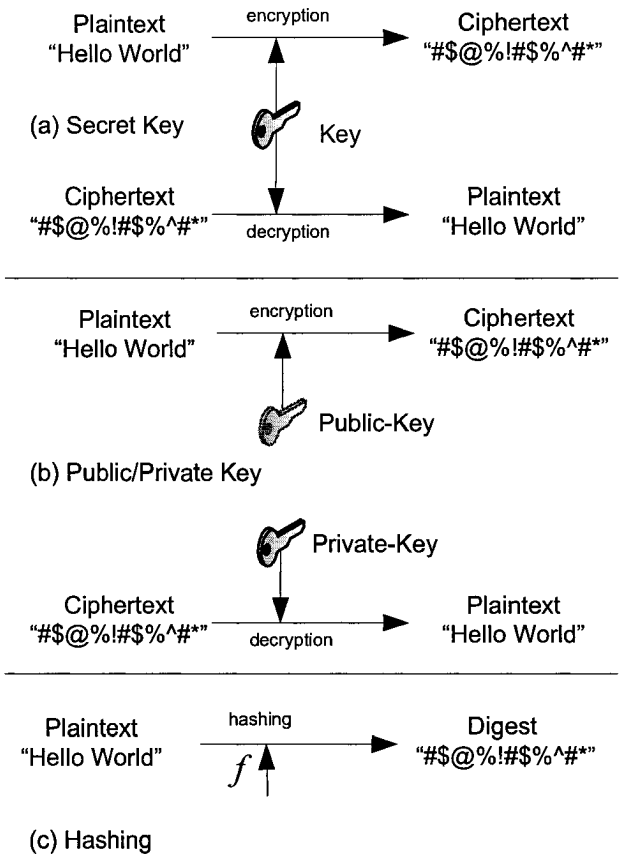


Fig. 1. (a) Secret key crypto (also known as symmetric key cryptography). (b) public/private cryptography (also known as asymmetric cryptography). (c) Hash function.

Public/Private-key cryptography has interesting properties which broaden its usability and applications. When a principal applies its own private-key to a message, other principal could verify that the message has not been changed by applying the public-key of the former which will lead to the original message. This operation is called *signing* and is very useful for proving *integrity*. Public-key cryptography is based on *modular exponentiation*. Without getting into its mathematical details, we will describe how the famous public/private cryptographic scheme RSA [41] works. First, we choose two large prime numbers p and q then multiply them to get n (i.e. $n = pq$). To generate the public-key, we choose a number e that is relatively prime to $\phi(n)$ where $\phi(n) = (p-1)(q-1)$. Now the public key is $\langle e, n \rangle$ and the private-key is the multiplicative inverse of $e \bmod \phi(n)$, denoted as $\langle d, n \rangle$. To get the ciphertext c for the plaintext m, we

do $m^e = c$. To decrypt the message, we just need to $c^d = m$. This works because $c^d = (m^e)^d = m^1 = m$.

2.2. Public-Key Infrastructure (PKI)

Public-Key Infrastructure (PKI) is a mechanism that offers the necessary components to generate, distribute, and control public-keys securely and confidentially (e.g. [5], [6]).

An ideal PKI [41] consists of the following components:

1) *Certificates:* A certificate is a digitally signed statement binding the key-holder's (i.e. principal's) name to a public-key issued by a trusted entity. However, the notion of a certificate has been expanded to include attributes other than key-holder such as nickname, email, account number, address, photo, or a combination of any other attributes [64].

2) *Certificates Authority (CA):* This is a repository and trusted entity that issues and signs certificates. The public-key of the CA is known and trusted by all nodes so that a node can verify any signed certificate issued by that CA. Certificates can be stored in any convenient location, such as the directory service [41].

3) *Revoking Scheme:* In some occasions, valid certificates need to be revoked due to business and/or security reasons. Usually revoked certificates are published in Certificate Revocation List (CRL). There exist some disputes against CRL and this was a point of controversy [64][48]. However, CRL is still used widely in various PKI implementations.

4) *Certificate Evaluating Method:* This is an evaluation scheme to validate a certificate chain of public-keys in advance that are known and trusted by their target name [41].

Even though some deployed PKI systems may leave out some components such as revocation, they are still called PKI systems, and this will not affect this discussion.

2.3. Dissemination of Revoking Information: CRL and Validation Schemes

Usually, every certificate has an expiration date that varies between few minutes to few years. On some occasions, it is necessary to revoke an unexpired certificate. If a principal's private-key is exposed, the principal is no longer under the administration domain of a given CA, or some of the certificate properties associated to that principal need to be changed are some examples that necessitates existence of a revocation mechanism.

CRL is a signed list of revoked unexpired certificates [41]. This list is distributed periodically to interested principals. CRL is a negative statement about the validity of the certificates it contains and, hence, any certificate that is not in the CRL is considered valid (a double negative). The problem of efficiently distrusting CRL has been addressed by researchers (e.g. [49],[52]). CRL could be eliminated by introducing a component dedicated to answering queries regarding certificates status.

The following list briefly illustrates the most common CRL distribution and certificates validation mechanisms:

1) *Basic CRL:* CA issues a list containing the serial numbers of revoked but unexpired certificates which contains the next issuance date. All revoked unexpired certificates will be listed again when the time of the next issuance occurs.

2) *Delta CRL:* This was introduced to reduce the bandwidth and redundancy associated with sending a complete list every time an update occurs. Instead, a delta CRL contains the changes from the last baseline CRL. Therefore, it is usually short and often contains no certificates [41].

3) *First Valid Certificate:* The scheme goal, again, is to make the CRL as small as possible and to allow certificates not to have a predetermined expiration time when issued. When the size of the CRL becomes too large, the CA will issue a notice requesting any principal which holds a certificate with a serial

number less than n to reissue a new certificate within some time from the date of that notice. The number n is chosen in a manner that only few serial numbers in the current CRL are greater than n. Revoked certificates with serial numbers greater than n must continue to appear in the new CRL [41].

4) *Online Revocation Server [4] (OLRS):* An OLRS could be a dedicated server that can be queried to check the validity of individual certificates. Typically, the verifier will query OLRS to check the status of the certificate in question online. However, if the OLRS is compromised, the worst the adversary could do is to revoke some valid certificates, given that the OLRS and CA use different keys. A scheme proposed by Micali [49] overcomes the shortcomings found in OLRS caused by cryptographic overhead by using the public-key every time a status reply is issued.

5) *Recertification:* Mukkamala *et al.* [52] present a technique to increase the validity of certificates without increasing the burden of revocation by introducing another entity called Recertification Authority (RCA). The CA issues a certificate with a long validity period; however, the principal is required to recertify at certain intervals from the RCA. This technique will prevent the CA from reissuing short-lived certificates.

6) *Counter Certificate:* In situations where revocation is a rare event, the CA could sign a counter certificate to revoke another certificate. However, the counter certificate should be made known to all interested parties.

2.4. Secret Sharing

Secret sharing (n,t), first indicated by Shamir [66], is a set of n random shares for a secret s such that it is possible to reconstruct s by the knowledge of t shares where $t \leq n$. Moreover, nothing could be derived about s by the knowledge of less than t shares.

Shamir's scheme is based on polynomial interpolation. Given t points in the 2-dimentional plane $(x_i, y_i) : i \in \{1,..,t\}$ and x_i's are distinct, there exists one and only one polynomial $p(x)$ of degree $t-1$ such that $p(x_i) = y_i$ for all i. Thus, if we pick a random polynomial $p(x) = a_0 + a_1 x + ...a_{t-1}x^{t-1}$ which has the degree $t-1$ and we assign $a_0 = s$, and evaluate: $q(1) = s_1,..., q(i) = s_i,..., q(n) = s_n$. By interpolation, we can find the coefficients of $p(x)$ and evaluate $s = p(0)$ to retrieve the secret. Knowing $t-1$ or less is not sufficient to calculate s.

2.5. Threshold Cryptography and Proactive Secret Sharing (PSS)

In the previous section, we showed the basic scheme for secret sharing. However, sometimes it is desirable to hide s even from the principals (e.g. servers) themselves. Y. Desmedt [26] showed how threshold cryptography can be used to perform cryptographic operations such as signing or decrypting a message without exposing the private-key of the service.

Here, we will give an example illustrated by Desmedt [26] to share RSA such that t servers can sign messages independently and then the combiner will obtain the complete signed message from those partially signed messages.

As a reminder, a cryptosystem is represented by a function that takes two inputs, *key* and *message*, and the key provided as parameter, e.g. $f_{key}(input)$. In this example, it is more convenient to make the key as an input so we have

$$f_{key}(input) = g_{input}(key), \quad (1)$$

where f and g are cryptographic functions.

An important property for a threshold cryptographic function to be shared practically is to be homomorphic, i.e. ,

$$g_b(k_1 + k_2) = g_b(k_1) * g_b(k_2), \quad (2)$$

where k_1, k_2 are keys and b is input.

Therefore, signing in RSA scheme will have $g_b(d) = b^d \bmod n$, where b is typically the digest of the message, d is the private-key, and n is the public modulus.

Since RSA secret sharing scheme is homomorphic, we have

$$key = \sum_{i \in B} (C_{i,B}) \cdot (share_i), \tag{3}$$

where B is a subset of the participants such that $|B| = t$ and $C_{i,B}$ is constant.

Thus, using (3) and (2) we have:

$$\begin{aligned} g_b(d) &= g_b(\sum_{i \in B} (C_{i,B}) \cdot (d_i)) \\ &= \prod_{i \in B} g_b(C_{i,B} \cdot d_i) \\ &= \prod_{i \in B} (g_b(d_i))^{C_{i,B}} \\ &= \prod_{i \in B} (b^{d_i})^{C_{i,B}} \end{aligned} \tag{4}$$

As we have seen in (4), a server that holds a share d_i will sign a message using its partial-key and sends the results with its identity (i.e., i) to the combiner. If the combiner has received sufficient responses (i.e. t), and by knowing $C_{i,B}$, the combiner will use (4) to sign or decrypt the message.

Another issue that should be considered in threshold secret sharing schemes is robustness against the case of one or more shareholders sending erroneous partial results. Again, as Desmedt [26] has pointed out in his survey paper, some schemes can be used to recover/discover erroneous results without exhaustive search.

Threshold secret sharing has some desirable properties [43] in MANET which make their combination worth noting:

1) *Distributed Control and Security:* A key feature in MANET is the absence of centralized control and the distribution of system management of the system is

consistent with the nature of MANET. To satisfy the ubiquitous nature of MANET, distributing the secret to various nodes would increase availability.

2) *Security vs. Availability:* The threshold t is the balance point between service availability and security. An adversary group must destroy $n-t+1$ share holders to turn off certification services, and break t servers to steal the secret.

However, secret sharing by itself does not defend against mobile adversaries [74],[57] in which the adversary moves to a node after compromising another for a limited time doing similar damage. By the time the adversary gains access to t nodes (t shares), the whole system is considered compromised. To enhance the security of the system, a proactive approach has been introduced by [39] to address this problem. Principals should refresh their shares to the same secret periodically in a way that old shares cannot be combined with the new ones to build the secret. The time between two refreshments is called *window of vulnerability*, and for the adversary to take control of the system, it must compromise at least t principals no later than one window of vulnerability period.

A brief explanation by [26] shows how to refresh the shares in homomorphic secret sharing schemes as follows:

If $(s_1, s_2, ..., s_l)$ is a share assignment for the key k and $(s'_1, s'_2, ..., s'_l)$ is another uniformly random share assignment for the "key" 0 (zero), then we have $(s''_1, s''_2, ..., s''_l) = (s_1 + s'_1, s_2 + s'_2, ..., s_l + s'_l)$ for the same key k.

However, if we restrict the process of generating shares to one principal, the system will have a single point of failure if that principal got compromised since it could generate wrong shares. Hence, t correct servers should generate their shares $(s'_{j,1}, s'_{j,2}, ..., s'_{j,l})$ where $j \in t$ and exchange the shares with all servers. Thus, the resulting share will be $s''_i = s_i + \sum_{j \in B} s'_{j,i}$, and it is an independent new share for the same key as depicted in figure above. However, each shareholder needs to prove that its contribution

$(s'_1, s'_2, ..., s'_l)$ is indeed a share of zero, since compromised principals may send bogus shares or do not even send shares at all! Luckily, verifiable secret sharing (VSS) schemes could be used such as [61] in which the principal can verify and testify to the correctness of the corresponding shares without disclosing them by adding some other public information.

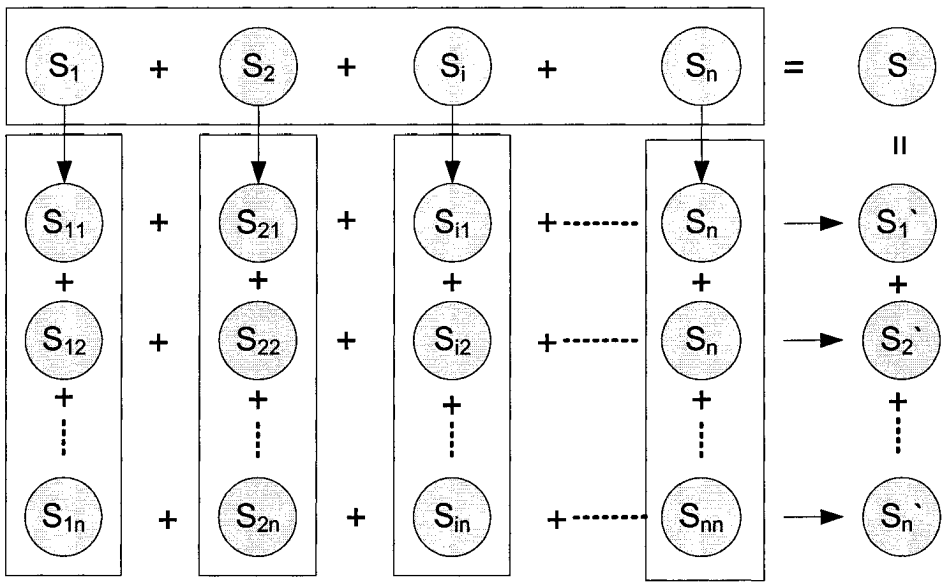


Fig. 2. Key refreshing sketch. Each server will generate shares for a random polynomial for the key *zero* (grouped by dotted line). Those shares are sent to corresponding servers and each one will combine the shares it received to construct its new share.

2.6. Overview of Mobile Ad Hoc Network (MANET)

MANET is a network architecture that can be deployed rapidly without any required fixed networking infrastructure. The nodes in MANET model share some unique characteristics such as mobility and joining/leaving the network very frequently, resulting in rapid changes to its topology. Moreover, nodes in MANET usually have constraints on their computation, storage, and power capabilities [32].

MANET is useful in situations like rescue missions, fast establishment of military forces, and national crises when the underlying infrastructure is no longer an option due

the damages done by natural or other disasters. In addition, MANET has some interesting commercial usages in which it could be more feasible than wired networks especially in short-term occasions such exhibitions, conferences, or festivals [32].

In MANET, nodes could be any entities such as soldiers, rescue team members, planes, trains, and ships which may possess significant variations in node speeds and moving patterns, ranging from stationary nodes to aircraft speeds, and from predetermined paths such as highways to unpredictable path such as rescue helicopters maneuvering around a disaster area. Even with such extreme conditions, MANET is expected to deliver a variety of traffic types including delay/bandwidth sensitive applications such as voice or video [32].

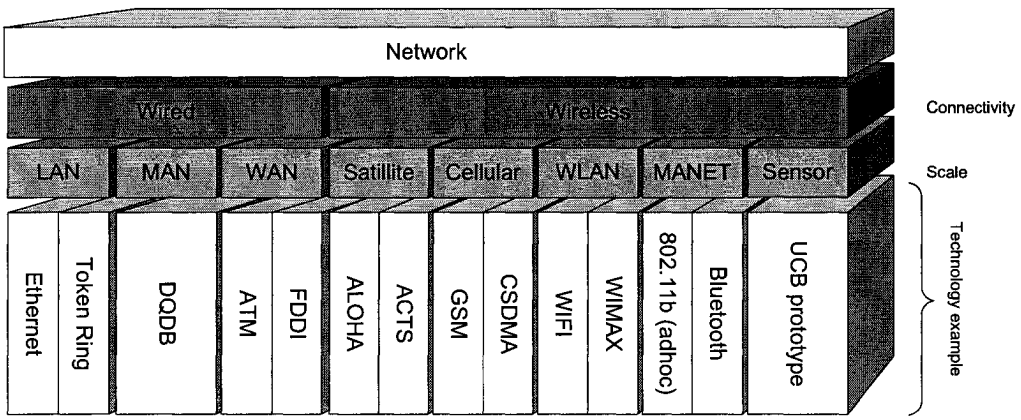


Fig. 3. Taxonomy of computer networking.

Moreover, MANET operates entirely through wireless medium and lacks centralized entities found in traditional cellular wireless networks such as base stations, MSC, HLR, and VLR (i.e. Mobile Switch Center, Home Location Registers, and Visitor Location Register respectively). Those entities are usually connected through hi-speed wired links and the wireless is used at the edge only (i.e. between the end-user and the base station).A wireless sensor network [7] is another more restricted environment with different assumptions which shares some of MANET constraints. Nodes are tiny devices deployed massively in a given place with no identification [56]. In Fig. 3, we represent a

general classification for networking systems to show where MANET fits. An example of current implementation of MANET includes IEEE 802.11 [2] and Bluetooth [1].

2.6.1 MANET characteristics

To distinguish MANET from its siblings, some assumptions that characterize MANET are listed here [32]:

1) *Power Source:* Nodes are equipped with portable communication devices and usually powered by batteries. Thus, limited battery life could restrict the transmission range, communication activity and computational power of the node.

2) *Transitivity:* Connectivity between nodes is not transitive; i.e., if node A can communicate directly with node B and node B can communicate directly with node C, then A is not necessarily able to communicate with C directly[1].

3) *Fixed ID:* All nodes are identified by fixed IDs such as IP addresses[2].

4) *Locality Assumption:* Usually, a large portion of the traffic will be between geographically close nodes although it is not prohibited to establish a connection among distinct nodes.

Thus, MANET imposes some requirements in design such as a robust routing and mobility management algorithms to increase reliability and availability. Moreover, any protocol or algorithm designed for MANET should have low overhead to reduce power consumption.

2.6.2 Routing in MANET

MANET introduces many challenges in designing reliable routing protocols due to frequent topology changes, power limitations, computation capabilities, and bandwidth.

---

[1] For example, transitivity holds in Ethernet since all nodes share the same medium.

[2] Some models, such as sensor network, do not require nodes to have an ID. Therefore, this assumption is not a luxury.

Those limitations impose a theme in MANET routing protocol design to favor algorithms which are low in overhead, adaptive, and robust [32].

MANET routing protocols could be classified as proactive or reactive [32]. Proactive protocols continuously exchange topological information among network nodes to keep routes as accurate as possible. Thus, all routing information is available immediately for any given node. Some example of such protocols include Optimized Link State Routing Protocol (OLSR) [37] and Wireless Routing Protocol (WRP) [53]. On the other hand, reactive protocols such as Dynamic Source Routing protocol (DSR) [16] and Ad hoc On-demand Distance Vector Routing protocol (AODV) [63] establish route on-the-fly and they do not attempt to maintain up-to-date topology of the network. Instead, they establish routes only when necessary. A hybrid approach also has been proposed, for example, a node in Zone Routing Protocol (ZRP)[62] proactively maintains topology to close neighbors and acts reactively for nodes that reside far away.

Another classification for MANET routing protocols is based on whether they are location aware[3] or not [32]. Examples of location aware protocols include Locating-Aided Routing (LAR)[42] and Distance Routing Effect Algorithm for Mobility (DREAM)[12].

2.6.3 Ad hoc On-demand Distance Vector (AODV)

The previous section gives a general overview of routing protocols found in MANET. This section will give expanded overview of the AODV routing protocol because we have used it in our work as the network layer in our simulation.

AODV is a reactive protocol that could be viewed as a variant of the well known distance vector routing protocol [67]. The AODV objectives are to broadcast discovery packets only when necessary. Neighbors exchange local broadcast packets called hello messages. The routing tables of the nodes are organized to optimize response time for local node movements and new route establishments. AODV assumes links are

---

[3] They could be equipped with an external device such as GPS to obtain their geographic location.

bidirectional. It uses a broadcast route discovery mechanism similar to DSR. The path discovery process in AODV is initiated when a source node wants to establish a connection to a destination node and no route is cached in the source table. It sends a route discovery packet (RREQ) to its local neighbors. The RREQ contains the source address, source sequence number, broadcast id, destination address, destination sequence, and hop count. Source address and broadcast ID fields uniquely distinguish a RREQ. Broadcast ID is incremented every time the node sends a new RREQ. Each node that receives a RREQ message either sends a route reply (RREP) if it knows a route to the destination, or it rebroadcasts the RREQ after decrementing the hop count (i.e. expanding ring search procedure). However, if a node receives duplicate RREQs from different neighbors, then it should forward only the necessary ones. After broadcasting the RREQ, it creates an entry in order to implement the reverse path setup that contains destination IP address, source IP address, broadcast ID, expiration time for that entry, and source node's sequence number. When the RREQ arrives at a node (possibly the destination), it checks whether the RREQ was received over a bi-directional link. Then, if the node has a route entry to the destination, it compares the destination sequence number of the RREQ with the one it has. If the sequence number it has is less than the one it just received, it should rebroadcast the RREQ. Otherwise, it replies its cached route back to the sender to establish the forward path [63].

2.7. MANET Security

2.7.1 Security Difficulties in MANET

L. Buttyán and J. Hubaux [18] discussed security issues in MANET and they refer to the following reasons to justify why it is difficult to secure such network:

1) *Channel Vulnerability:* Wireless channels are vulnerable to eavesdropping and the injection of faked messages without any physical access.

2) *Node Vulnerability:* By not residing in a physically protected place, nodes can be captured and compromised easily by attackers.

3) *Absence of Infrastructure:* Classical security methods that require an infrastructure are no longer useful due to the distributed nature of MANET.

4) *Dynamic Topology:* This is a fact in MANET, and it makes routing difficult since incorrect routing information can be generated easily by compromised nodes making it difficult to distinguish this case from legitimate topology changes caused by mobility.

2.7.2 Availability

Availability has been addressed from different perspective and MANET is vulnerable to DoS attacks by abusing communication protocols which lead to a reduction of honest node performance or clogging the access to a needed server.

If we take the MAC layer protocol of 802.11 [2] as an example for typical MANET, nodes exchange RTS and CTS packets to reserve the channel before transmission. However, a node transmitting a packet must backoff for randomly selected period *b* which is between 0 and CW (Contention Window). If collision happened, CW is doubled. Misbehaving nodes can take the advantage of the algorithm to increase their throughput by not respecting the protocol and choose small values to shorten the backoff time and increase their probability of getting the channel. This problem has been stated and studied by P. Kysanur and N. Vaidya [44], and they developed a solution to detect malicious behavior by using a traffic analysis scheme.

Another approach, which applies game-theoretic concepts, considers the following dilemma: Cooperation will consume resources, and it might result in a disadvantage, but if all nodes did not cooperate, the result is bad for all nodes. This dilemma was the basis for designing Cooperation Of Nodes, Fairness In Dynamic Ad-hoc NeT-works (CONFIDANT) protocol [17]. Each node will learn and observe misbehaving nodes by monitoring the observations and reputation records generated by others. For example, if a node believes that another is misbehaving, then it can take action in terms of packet forwarding to avoid routing through that node, or it can report this behavior to another node by sending alarm messages about the node in question. The authors showed,

by using simulation models, that CONFIDANT can cope well even when half of the nodes are misbehaving.

2.7.3 Secure Routing

Originally, MANET routing protocols like DSR [16], AODV [63], OLSR [37], WRP [53], ZRP [62], LAR [42], and DREAM [12] were examples of classic MANET routing protocols that have been proposed to be robust against topology changes, but none of these have applied measurements to defend against attacking route information [73]. A malicious node can learn the network topology by monitoring routing messages to conduct an attack that brings the network down. For example, it can inject bogus routing information, reply old information, or distorting routing information [73]. Defending against routing attacks is hard because it is difficult to tell if invalid routing information is caused by a topology change or a compromised node. Another interesting type of attack is a "wormhole" attack (Y. Hu and *et al.* [33]) in which attackers tunnel routing and other traffic in one location and release them in other part of the network which damages the topology. The authors proposed a solution to this type of attack by the use of "packet leashes" in which geographical and/or temporal information are included, allowing the receiver to determine if the packet has traversed a realistic distance or not.

Secure-AODV [72] has been introduced as an extension to AODV routing protocol to protect the routing protocol's messages. Every node is assumed to have a certified public-key for all other nodes. The sender of any routing message will sign the packet and include some information generated by hashing function in a way that guarantees no intermediate node can decrement the hop count to solicit traffic by advertising a shorter path and other solutions (e.g. [59], [33], [34], [65]) which aim to provide solutions to various routing security problems.

2.7.4 Intrusion Detection

Encryption and Authentication are considered as intrusion prevention mechanisms. However, sometimes nodes get compromised, and their private-keys are exposed, which means the system is no longer secure. Thus, intrusion detection measures are required to overcome this problem. In high-security networks, Intrusion Detection Systems (IDS) are

necessary as a last line of defense [75]. However, classic IDS are inadequate for MANET since they rely on real-time traffic analysis for packet traces collected from switches, routers, and gateways. Unfortunately, MANET lacks such infrastructure, and there are no physical boundaries for the system. Thus, the only way to collect data is to let nodes monitor traffic, which takes place in their radio range. MANET, due to its dynamic nature, requires an IDS solution to be fully distributed, be able to analyze local audit data, and make sure that it can distinguishes the thin line between anomaly and normal behavior. Y. Zhang and W. Lee [75] [76] addressed these issues by developing a new architecture called Distributed and Cooperative Intrusion Detection and evaluated its ability to detect intrusion. Each node has an IDS agent, and those agents collaborate to secure the system. Using simulation, the system has been evaluated to detect routing attack, packet attack, and no attack to measure false alarms. Their simulation shows very useful results and detected 99% of the attacks with less than 1% of false alarms.

2.8. Summary

This section has presented necessary background related to our work. A general overview of encryption and public/private-key cryptography has been shown followed by an overview of PKI and a review of its main components. We dedicated a section to discuss revoking schemes and gave brief details about each one. A description of the Shamir scheme of secret sharing has been given with a detailed example. We followed that section with a description of threshold cryptography and how to make it practical using proactive secret sharing techniques. Another section was devoted to MANET and its importance, challenges, and characteristics. Taxonomy of routing protocols in MANET was given with a special focus on AODV protocol. We continued our discussion on security problems in MANET and gave examples of availability problems, secure routing, and intrusion detection techniques. We will delay discussing solutions for providing general public-key services until the next section as those are considered related work.

SECTION 3
RELATED WORK

The aim of this section is to discuss the attempts to adapt public-key solutions into MANET. We will begin with a proposal to secure closed-networks in Section 3.1. Section 3.2 is about certificate-based security for self-organized networks. Sections 3.3, 3.4, and 3.5 are dedicated to discussing URSA, COCA, and MOCA respectively, which are designed to address the problem of applying *online* CA into MANET. Finally, concluding remarks are in Section 3.6.

3.1. Closed-Network Solutions

Closed-Network is a kind of network in which the number and the identity of participants are known a priori. The benefit of such a classification is that it allows security features to be implanted or agreed upon in advance since all information is known. For example, Asokan and Ginzboorg [11] have proposed a solution for establishing a secure ad hoc network without access to a public-key infrastructure. They gave a scenario of a small group of people (each one with a laptop) meeting in a classroom and wanting to communicate securely. To achieve that goal, they need to agree on a password and use it for that session (e.g. write it down on the blackboard).

Hu *et al.* [34] proposed more *automated* ways to do secure routing with public-keys by preloading every node with the public-keys of all other participants as part of the initialization phase for their proposal. The problem of such an approach is flexibility since new nodes cannot be added to the network. They have suggested another modification by allowing an *offline CA* to certify nodes; therefore, nodes can authenticate each other by certificates signed by that CA.

3.2. Self-Organized Solutions

The term Self-Organized security is used to refer to a setting where there is no central authority to control the establishment and/or releasing of security associations and it is up to each node to decide which other node(s) to trust or communicate with [20] by making each one act as a self-CA.

The first solution for the self-organized public-key management system for MANET which is similar to PGP [41] was suggested by Hubaux *et al.*[36]. Users issue certificates to each other based on some kind of contact such as personal, social, friendship, relational or association (represented in each node by a directed edge in the node trust graph). Each user maintains a local certificate repository, and when two users want to verify each other, they exchange their local repositories (i.e. their digraphs) and try to find appropriate certificate chains connecting both ends in the resulting merged graph. Hubaux *et al.* have suggested an efficient method, The Shortcut Hunter algorithm, to find the shortest path between any two principals (i.e. nodes). The framework assumed that nodes are honest and will not issue false certificates, but the authors have suggested some metrics when that assumption is relaxed.

3.3. Ubiquitous and Robust Security Architecture (URSA)

The goal of Kong's *et al.* [43] is to provide pervasive CA services by making all n nodes in the network share CA functionality. All participants sharing the private-key is obviously unfeasible and not realistic since $t = n$ (i.e. only one node needs to be compromised to deny the service). To overcome this problem, the same private-key is distributed among coalitions (as referenced in URSA), and each one has at least t member nodes, which are one hop apart from each other. Thus, a client needs to contact t nodes only, rather than n, by sending a local broadcast request and collecting t replies to obtain the service.

This scheme has an adapted RSA-based design for cryptographic operations. The CA service has a single private-key for signing certificates and it is not known, as a whole, to any member node of the MANET. However, any t nodes at a given coalition

can collectively act as a CA by using their shares to sign certificates and every node has its own public/private-key to do end-to-end security functions (e.g. encrypting or signing messages as an individual).

Nodes obtain their shares either when bootstrapping the system or using self-initialization algorithm. At the startup of the system, a centralized management, also known as "trusted dealer," will distribute t shares to t nodes just before forming the MANET. After initializing those t nodes, the centralized management is no longer needed, and any additional uninitialized node can retrieve its secret share by sending a one-hop local broadcast to a coalition of t nodes. They then send their partially signed certificates back to the requester. Once the requester collects t partial certificates, it reconstructs the complete certificate. However, once a certificate is issued, it is trusted by the entire; network, therefore, a well defined issuing policy is needed to control the issuing of certificates.

Certificate renewal must be done before its expiration time by presenting a valid unexpired certificate to a coalition. Then, the coalition will check its CRL, verify the certificate, and issue a new certificate with a new, later, expiration time.

A certificate is implicitly revoked when it is expired and explicit revocation is handled by an issuance of a counter certificate that flooded into the network. Besides flooding, neighbors can safely exchange their local CRL cache to build a more complete CRL.

From a security point of view, the adversary group will not have the secret if less than t nodes are compromised. Proactive Secret Sharing (PSS) protocol is used to prevent the adversary from learning the service private-key in the long term with the assumption that it cannot compromise more than $t-1$ between any two consecutive secret-sharing updates. Additionally, it is easy for a client to verify whether the reconstructed certificate is correct or not.

URSA assumes that t one-hop neighbors or more usually exist. Nevertheless, in a mobile environment, this may not be the case, and Kong *et al.* suggest that if the client

does not receive sufficient response, within some time limit, then it should move to another location and try another coalition.

The authors provided Unix and NS-2 [47] implementations for URSA to measure cryptographic and mobility costs respectively. They simulated network topologies, which have 30 to 100 nodes and mobility between 1 to 20 m/sec for 5 minutes using a random waypoint. The value of t was chosen to be 3 when simulating a 30 node scenario and 5 otherwise. The success ratio is almost 100% for all variations.

3.4. Cornell Online Certificate Authority (COCA)

COCA [74] is a secure online certification authority that was originally designed and deployed in LANs and the Internet. The designers made weak assumptions[4] regarding the environment in which COCA can operate. No assumption has been made about execution speed and message delivery delays. COCA applies some traditional means to defend against denials-of-service attacks such as processing authenticated requests only, caching results of expensive cryptographic operations, and multiplexing resources to balance the demand of various classes of requests.

COCA is the first system that integrates a Byzantine quorum system [46] with threshold cryptography and addresses some problems that arise when combining the two. Additionally, the authors have demonstrated experimentally the difficulty of launching successful attack against their system.

A certificate request sent by a client is received by a selected COCA server, called the delegate, which forwards the request to all COCA servers. Then, it waits for signed partial certificates from a quorum of COCA servers. Once those partials have been received by the delegate, it invokes COCA's threshold signature protocol to sign that

---

[4] i.e. the protocol, theoretically, will converge to a stable state regardless of how long it takes to go to that state. One such weak assumption is that a correct message will eventually be delivered when sent repeatedly.

certificate. To update certificate information, the delegate will need further steps, which are discussed in more detail at [74].

It is possible that the delegate itself can be compromised and deviated from the protocol. Thus, the client needs to contact α other delegates to ensure that it will get at least one correct response. However, in normal circumstances, the client may contact just one delegate, $\alpha = 1$, to do certificate operations. Zhou and Hass [73] sketched how to deploy COCA in MANET, but they did not provide any performance results. The original COCA protocol was written in "C" and they [74] provided some measurements for a sample deployment of four machines with t set to one.

3.5. Mobile Certificate Authority (MOCA)

MOCA [70] was created taking a MANET environment into consideration as an extension to an earlier prototype [71]. A MOCA is a mobile node within the MANET selected to provide distributed CA functionality. Typically, more powerful and more secure nodes are chosen to provide CA service to other nodes. MOCA nodes apply threshold cryptography to share the responsibility and increase availability of the MANET. Client nodes use MP (MOCA certification Protocol) to contact sufficient MOCAs and use their services.

When a client wants to contact another client, it sends "Certificate Request" (CREQ) packets to t or more MOCA servers and waits for some time. Any MOCA receiving a CREQ will reply by sending a "Certification Reply" (CREP) packet containing its partial signature. Once the client receives t valid CREPs, it can reconstruct the whole certificate. CREQ and CREP messages are similar to Route Request (RREQ) and Route Reply (RREP) messages in on demand ad hoc routing protocols such as AODV [63] and DSR [16].

To increase system availability, MOCA clients send β unicast CREQ, where $\beta = t + \alpha$, and α is determined in each client by inspecting the status of the network. Moreover, to avoid flooding the network, clients inspect the route table to see if any

cached routes to MOCAs are available, thus, making flooding the last resort. Yi and Kravets [70] have tested several alternatives when the number of cached routes to MOCA servers is more than β.

Yi and Kravets' performance criteria were to measure the ratio of CREQ and CREP messages in their scheme over the sum of CREQ and CREP in pure flooding. MOCA scheme has achieved a 5-30% improvement over pure flooding. We will discuss their results in the next section.

It is worth mentioning that one minor technical difference between MOCA and COCA frameworks is that the construction of the certificate is done at the client side which adds CPU overhead to MOCA clients, while the reconstruction is done at the delegate server in the case of COCA framework.

Finally, we present an illustration to show the difference between some of the related work presented in this section.

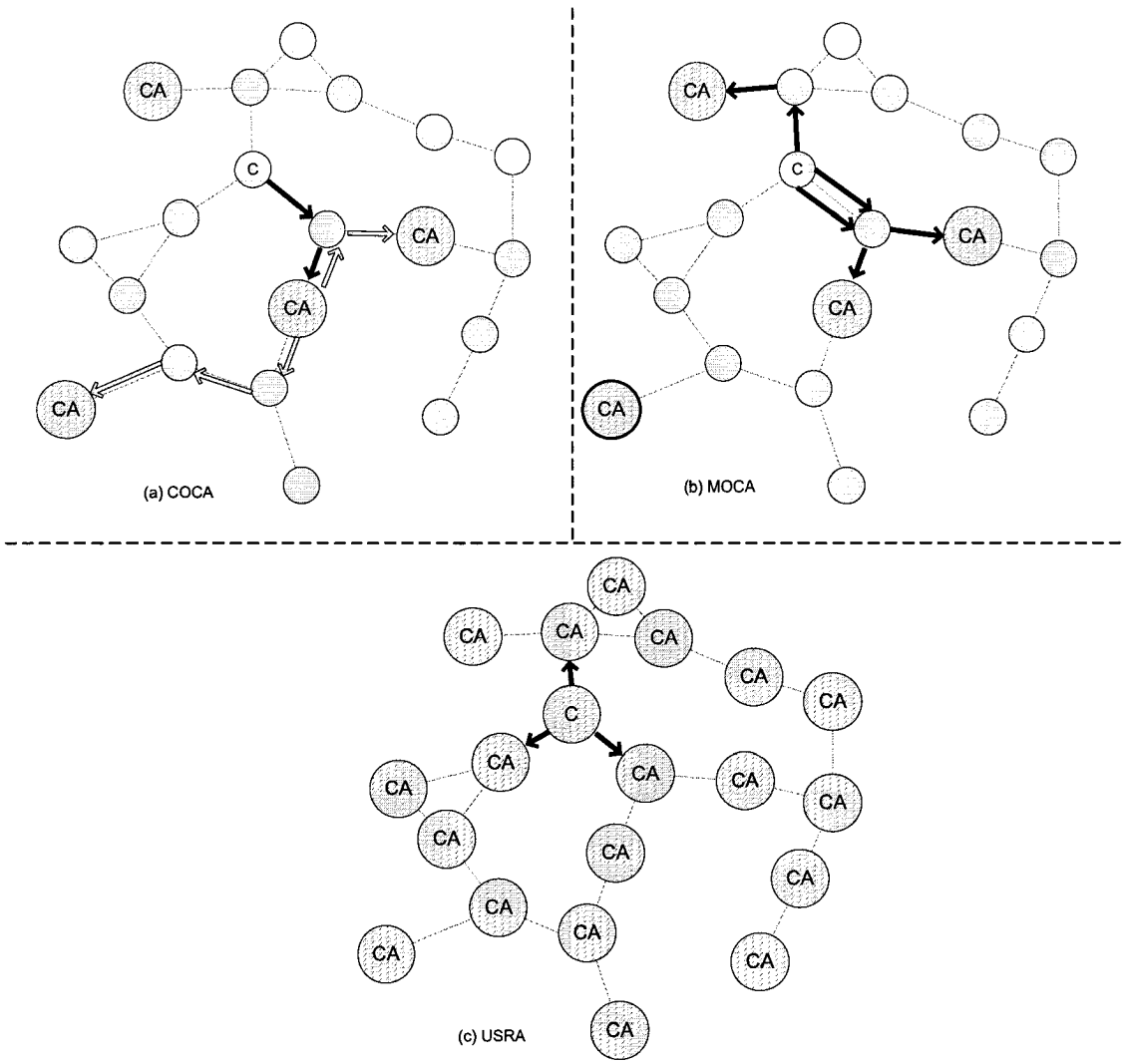


Fig. 4. A hypothetical request sent by a client (denoted by "C") to obtain a certificate from available CA servers (denoted by circle labeled "CA") using various protocols. (a) COCA (b) MOCA (c) URSA.

Fig. 4 and Fig. 5 show, respectively, how a request and its reply would be handled for a client to obtain a certificate for various protocols. Each packet sent is represented by a black arrow and the light gray arrows in Fig. 4(a) represent a communication made by the CA on behalf of the requesting client. We assume that the threshold is three (i.e. the client needs at least three correct replies).

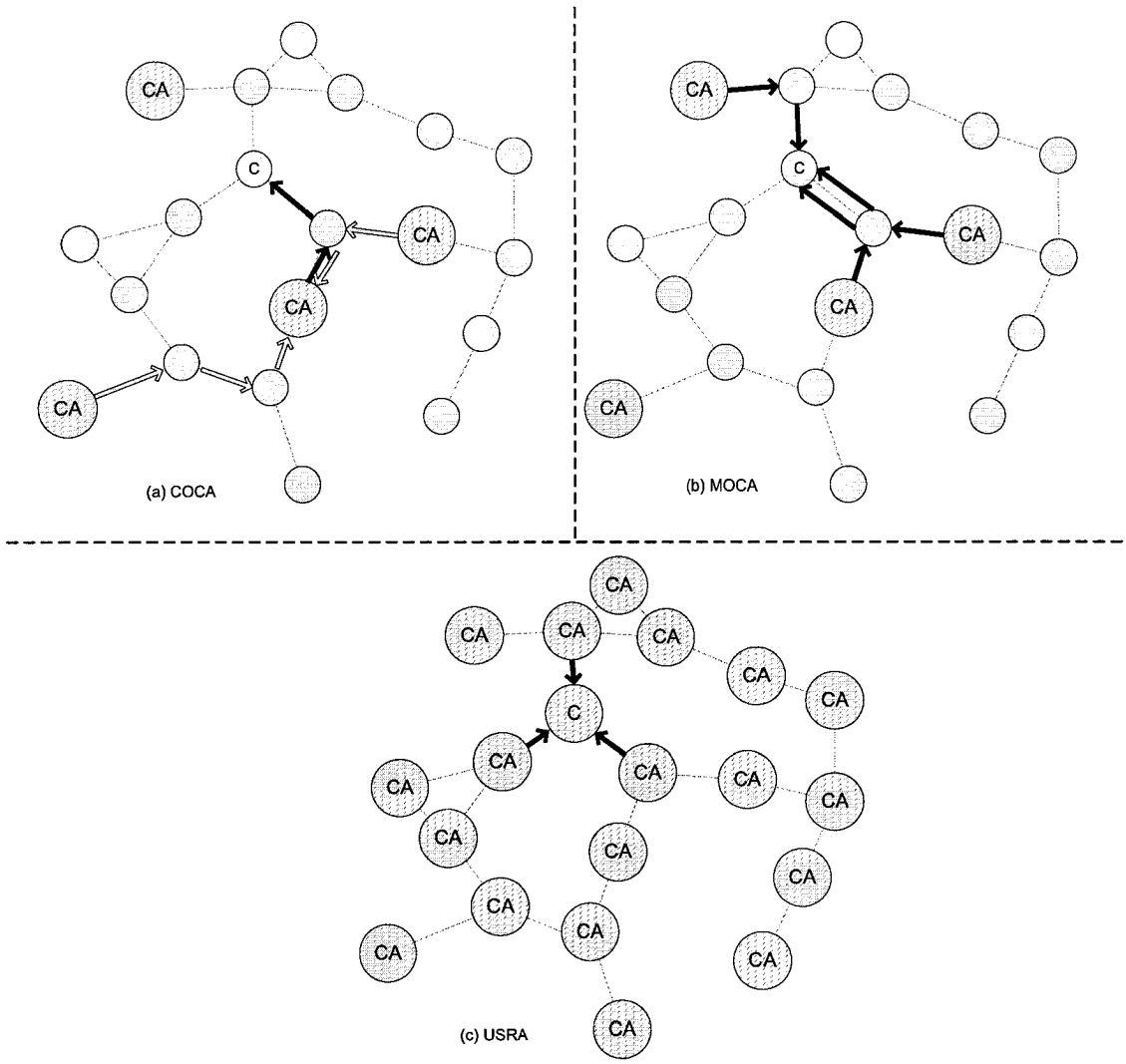


Fig. 5. The replies sent by servers (denoted by "CA") back to the requester (denoted by circle labeled "C") using various protocols. (a) COCA (b) MOCA (c) URSA.

3.6. Conclusion

This section was dedicated to discussing related work adapting public-key and PKI services into MANET. We have started the section by showing security solutions for a small MANET followed by an overview of a self-organizing key-management system. We have expanded the discussion to include USRA, COCA, and MOCA to show more relevant work to what we are proposing.

SECTION 4
CACMAN FRAMEWORK

This section is dedicated to describing our framework from the design perspective. Section 4.1 describes our motivation to develop "CAching Certificates in Mobile Ad hoc Networks" (CACMAN). The system assumptions are listed in Section 4.2. Section 4.3 describes the CACMAN-basic protocol in depth. Other possible techniques for implementing the CACMAN communication protocol are discussed in Section 4.4. Variations and enhancements to the basic framework are given in Section 4.5. Section 4.6 discusses CRL techniques and the consequences when deployed in the framework. Miscellaneous details such as key refreshing between CACMAN CAs and "partial certificates compaction and redundancy removal" are addressed in Section 4.7. Section 4.8 illustrates deployment scenarios in hybrid-MANET. Section 4.9 is our conclusion.

4.1. Motivation

Based on the discussion of related work found in Section 3, we strongly believe that there is room for enhancing the performance and service availability of the CA service. The main objective of this work is to reduce the amount of overhead (e.g. total number of packets per request) associated with providing pervasive *threshold based CA* services without significantly affecting its availability or compromising its security.

Closed-network (Section 3.1) and self-organized (Section 3.2) security solutions are feasible for small size networks, but they cannot scale when the number of nodes is large and/or the membership is dynamic. Moreover, self-organized solutions, though elegant, are based on the transitivity of trust security assumption (i.e. PGP-like) in which there is no agreed third entity (or anchor point) that every principal will automatically trust. Thus, self-organized solutions are suitable for securing personal communication at the application level as noted by Čapkun *et al.* [20]. Moreover, when a node joins the

network, it cannot start functioning unless it builds its own trust network, which takes a considerable amount of time.

The URSA scheme (Section 3.3) is interesting but vulnerable because small values for t were chosen to increase the availability of the system; however, this also jeopardizes its security. Furthermore, finding one-hop t neighbors to obtain the service is difficult especially when t is high. We believe that the tradeoff between availability and robustness can be separated. Another shortcoming is that any issued certificate by any coalition is trusted by definition, which is problematic since there is no known entity to make the decision of issuing or revoking a certificate. Moreover, there is considerable overhead for each node when joining/leaving a coalition every time it changes its location. Besides, it is not known what a coalition member should do if the number of the members are less than t.

On the other hand, MOCA framework (Section 3.5) is useful but very expensive. More specifically, we have major two concerns about MOCA. First, the number of MOCA nodes is relatively high, about 10-20% of the total number of participants. Making such large number of CA servers to increase availability does not come without consequences, such as the overhead associated with replica synchronization and key refreshing between CAs, besides increasing physical exposure of the system. Notice that the flooding protocol [71] is independent of the number of CAs and, hence, is not sensitive to the number deployed in the field. Second, and more important, the ratio of MOCA Certificate Requests (CREQs) and Certificate Replies (CREPs) as compared to flooding is high- at best 70%- when $\beta = 5$ (remember $t \leq \beta$) and $n = 30$ [70]. In our opinion, making a vital service that expensive is undesirable. COCA (Section 3.4) shares the same shortcoming as MOCA.

The argument of Kong's *et al.*[43] against dedicated CA service (i.e. COCA and MOCA) makes sense. They argued that high mobility will make contacting dedicated CA servers in a timely fashion a non-trivial task. Besides, the CA could be multiple hops away from the client, which increases latency.

Our work, based on caching, could be viewed as an enhancement to overcome the performance concerns regarding MOCA framework. The novelty of such enhancement is to maintain availability and efficiency of the system even when t is high –more secure and less exposed to adversaries- without significant tradeoff to the network overhead by, collaboratively, caching full and partially signed certificates[5] into clients' memory. Our work could be viewed as a hybrid between MOCA and URSA with more novel and unique enhancements. Our protocol still needs CA servers, but also requires clients' participation to enhance the service.

Caching has been used in various aspects of computer systems, either as hardware or a software component, to avoid many inherent delay penalties associated with the high speed gap of various system components or to increase their availability in the case of a failure that happened to other components. For instance, in MANET, caching of route information has been used to reduce the overhead associated with flooding the network by route discovery messages (e.g. DSR [16] and AODV [63]). Another usage of caching in MANET is to increase the availability of web services such as in [31], [13]. In our case, we are trying to decrease the service overhead and power consumption associated with such overhead.

We believe that certificate caching is an important concept due to following factors:

1) *Small Foot Print:* A certificate consumes around 1k, and this is a negligible size compared to other objects (e.g. multimedia objects).

2) *Long Validity:* Usually certificates tend to have a relatively long life, about a year, and this is a desirable property since it increases the ability of clients to cache them for long time without the necessity to purge them due to a new update.

---

[5] In this section, the terms 'share' and 'partial certificate' will be used interchangeably.

3) *Centralized Update Point:* A certificate is usually updated by the CA only, and this eliminates many problems associated with replicated data objects when conflict update happens because the object is changed by different entities at different locations.

Caching certificates is safe, and if an adversary compromises less than t CACMAN CA servers and clients have cached the faked share generated by these compromised CAs, then the faked shares will not reconstruct correctly. This could increase the overhead for a client since it needs to identify the compromised shares. However, schemes discussed by Desmedt [26] and additional cautions could be applied like using misbehaver detection schemes (Section 2.7.2) to alleviate the problem.

4.2. Framework Assumptions

This section lists the most important assumptions and conditions that make CACMAN useful and practical. We would like to emphasize that many of these assumptions are found in related work.

1) *CA Initialization:* CA servers must be initialized by a trusted entity or some Verifiable Secret Sharing (VSS) techniques are used to ensure correct partial-keys have been deployed and every CA indeed receives a valid share of the secret.

2) *Window of Vulnerability:* Malicious node(s) cannot compromise more than t CA servers at any given window of vulnerability.

3) *CA Role:* CA in CACMAN is the same as in MOCA. We did not involve CA servers in any optimization or enhancement so that we can measure CACMAN clients' performance.

4) *CA Cost:* CA servers are expensive entities in the system and cannot be deployed massively (e.g., many CA operations are of the order $O(n^2)$).

5) *Node Correctness:* Nodes in CACMAN, at any given time, are either malicious or correct. If a CACMAN node is correct, it will follow the protocol. Otherwise, it will deviate from the protocol arbitrarily.

6) *Security Essentiality:* Security is an essential part of the network and certification services are used heavily in most services such as authentication, sending messages, exchanging routing information, and so on[6].

7) *Bandwidth Limitation:* The bandwidth and the delay in the underlying MANET in which CACMAN is deployed is limited, and we are more interested in trading storage to save some bandwidth.

8) *Fair Links:* is a communication channel exists by which if we send an enough messages between any two points, some will arrive correctly [74]. If no message can arrive intact, it is impossible to build a useful communication protocol. This assumption is reasonable and realistic for MANET.

9) *Power Consumption:* Sending a packet in MANET is an expensive operation in terms of power consumption (e.g. sending a 1 kb data packet will consume as much energy as processing 300 million instructions in some general purpose [60] or PDA [3] microprocessor.

10) *Open Network:* CACMAN nodes do not necessarily know or trust each other since assuming a self-organized or closed network will reduce the necessity of adapting PKI solution.

11) *Online CAs:* This assumption is to differentiate it from offline CA, as assumed in some other public-key solutions (Sections 3.1 and 3.2).

---

[6] Thus, a client may have more than one certificate, i.e. one for login and another for email. Therefore, the reader should not assume one certificate per client, although we have assumed that in the simulation for simplicity.

12) *Single CA Service:* This assumption is to simplify the discussion without losing generality. The performance evaluation for CACMAN has been done with this consideration.

13) *Servers Communication:* CACMAN does not make any assumptions on how servers are communicating with each other (e.g. multicast) for some operations that require consensus such as key refreshing or signing CRLs.

14) *MANET Assumptions:* To avoid redundancy, MANET model assumptions mentioned in Section 2.6.1 still hold in CACMAN.

4.3. Protocol Description

Two important operations occur in every CACMAN node, CREQ and CREP[7] which are described here. When a client wants to obtain other client(s)' certificates to establish a secure communication or to encrypt some messages, the following algorithm will be executed:

1) It checks its own cache to find out the availability of a complete certificate, or it will identify the missing parts if partially signed certificates are found.

2) If there is no sufficient number of partially signed certificates, then it makes a local broadcast. However, if it happens that the source has a short route to the subject in question, it should favor sending the CREQ to that subject instead of making a local broadcast.

3) Every client that receives a CREQ message inspects its cache for a possible hit. It sends a CREP back to the sender if a full or any partial certificate is found for the subject.

---

[7] We are using the same abbreviations, CREQ and CREP, used in [70], to refer to 'Certificate Request' and 'Certificate Reply' respectively.

4) The request initiator waits for a specified time; if it receives sufficient responses then it reconstructs the certificate. Notice that the client could receive a complete certificate, which eliminates the reconstruction step but not the verification. If the request initiator times out because insufficient partials have been received or the reconstruction has failed, then the client may try to contact CA servers by flooding the network or by sending unicast messages if it has sufficient routes to them. Otherwise, it reports a failure to the client or retries later.

The last step of the protocol is, somehow, *open* because, in our opinion, such decisions should not be taken without knowing the application domain.

Fig. 6 illustrates the state diagram for CREQ operation as described in the protocol.

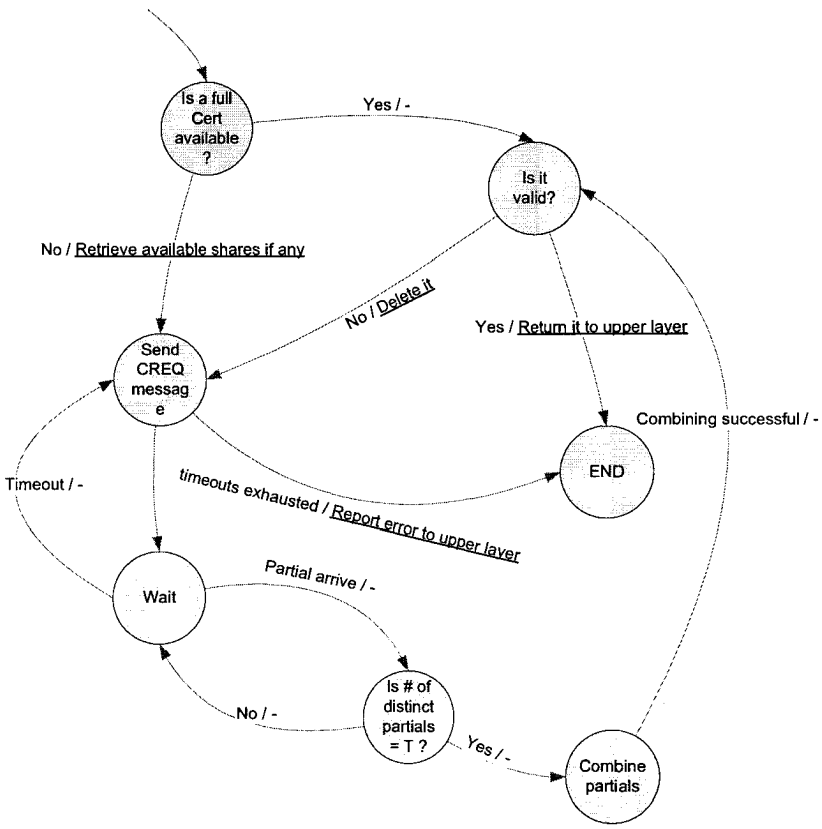


Fig. 6. State Diagram for CREQ operation in CACMAN. (Transition format is "cause / output")

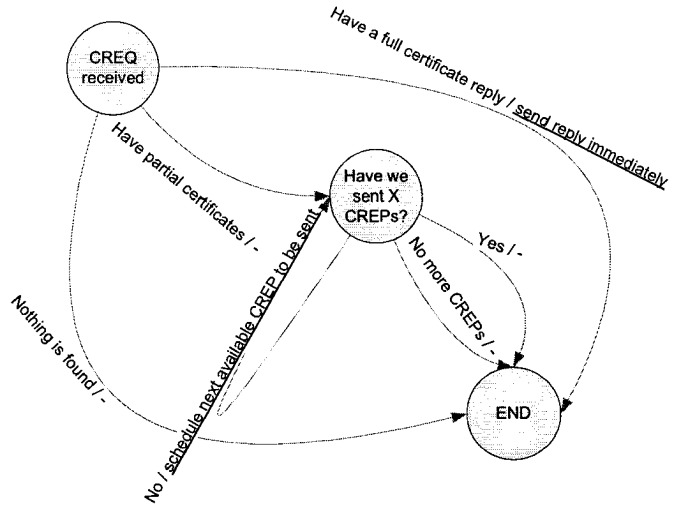


Fig. 7. State diagram for CREP (with X-Reps enhancement).

4.4. Alternative Communication Paradigms

In Section 4.3, we mentioned that a CREQ is sent by local broadcast. However, it does not mean that this is the only way of sending CREQs. Other possible alternatives to contact CA servers or CACMAN, in no specific order, are[8]:

1) *Unicast:* a client will contact every server (CA or CACMAN) by sending unicast CREQ messages.

2) *Flooding:* a CREQ will be flooded into the network and every server that receives the request will send a reply.

3) *Scoped Flooding:* a CREQ will flood the network but with controlled "Time to Live" (TTL) to reduce the amount of overhead and still guarantee the CREQ is received by a sufficient number of servers.

[8] Of course, these methods are generic, but we show them here in CACMAN context.

4) *Multicasting:* a multicast tree is built between a client and all servers. A CREQ will be sent to the tree and CREPs are sent back to the client. There is overhead associated with maintaining and updating the multicast tree.

5) *Scoped Multicasting:* The main idea is similar to multicasting. However, once a CREQ is sent, a TTL value associated with it is received by some CA servers and does not necessarily reach every node in the tree, especially leaf nodes.

The first and second method have been used in MOCA (Section 3.5) and the remaining three are examined by Carter *et al.* [22] but in the context of manycast problem. The performance of the first and second techniques are examined in Section 5 and compared to CACMAN. Carter *et al.* have shown that the forth and the fifth methods were as expensive as flooding[9]!

4.5. Variations and Enhancements

The previous section illustrated CACMAN in it basic form. Here, we will show some enhancements that could be incorporated into the protocol to increase its performance. Each of those enhancements, except the one in Section 4.5.4, has been simulated, and the results are reported in Section 5.

4.5.1 X-Replies

Basically, instead of sending one CREP when a CACMAN server receives a CREQ, it will send at *most* X different replies to the requester based on their availability if there is no full certificate found. The reason behind this enhancement is to increase the likelihood of getting all necessary shares to reconstruct the certificate and to make the chance of delivering more desired partials if the earlier one, though not identical, did not arrive due to congestion or disconnection.

---

[9] In that paper, the authors examined ADMR [40] as multicasting protocol.

4.5.2 Lazy-Replies and Lazy-Replies with Cancellation

"Lazy-Replies" and "Lazy-Replies with Cancellation" share the goal of reducing the amount of dropped packets caused by "reverse storm effect" [21] when packets arrive at the originator from multiple links at the same time. This is done by allowing CACMAN server to defer their CREPs, if they had any, once they receive a CREQ for a randomly chosen period between 0 and δ, similar to backoff algorithms used in various networking protocols. The exclusive goal for "Lazy-Replies with Cancellation" enhancement is to reduce packet overhead by allowing CACMAN client which had sufficient number of replies, i.e. t, to broadcast a cancellation packet, after their CREQ, notifying CACMAN server(s) that have scheduled their CREP(s) but not yet sent them to disregard their outgoing CREP(s).

4.5.3 Cache-Focus

The goal of this enhancement is to understand the effect of forcing some discriminative rulings on which partials are allowed to be cached in CACMANs' memory. Intuitively, we thought "Cache-Focus" would reduce the amount of duplicate packets sent by clients due to possible similarities as in the basic protocol. In Cache-Focus, each client, once it boots up, will choose one or more CA servers and cache any partials belonging to the chosen CAs. These partials could be obtained when the client receives CREPs by request or overhears them due to requests issued by other clients. Of course, a client will apply some cache replacement policy once it gets a candidate partial and its memory is full.

Finally, one drawback of Cache-Focus is that filling the cache will take longer with the CACMAN nodes because not every overheard or received partial is allowed to be cached; however, a slight enhancement could relax the problem by allowing partials which are not members of the focus group to be cached if the memory at the client permits, but those partials will have the highest chance of being victims if another candidate share arrives.

4.5.4 Explicit CREQ

The purpose of this enhancement is to eliminate duplicate partial replies if the requester already has those partials. For example, if the requester already has 13 partial certificates

out of 15 for a given subject, then the client is interested in acquiring any two partials that it does not have. There is no value of getting the same partials again, especially when they affect the bandwidth and energy, and consume valuable CPU time. One way of implementing this enhancement is to add bitmap fields in the CREQ header, in which each bit represents a server ID; the requester will set the bits that correspond to the partials it has for the certificate in question. Now, when a CACMAN node receives a CREQ, it can decide whether its cache will benefit the requester or not.

4.6. CRL in CACMAN

The National Institute of Standards and Technology (NIST) has requested that the MITRE Corporation study the cost of implementing and deploying PKI for the federal government [51]. The conclusion shows that Certificate Revocation List (CRL) is considered as the most expensive component of PKI framework. In MANET, the situation is more challenging due to its inherent properties. In this section, we will analyze the consequences of integrating various CRL schemes into CACMAN without favoring any specific scheme since this needs more detailed assumptions.

Let us first start with the simplest solution, that is, a system with short-lived certificates only. Such systems will require CA servers to be always online and no CRL needs to be issued. The consequence of this model is that it will destroy the temporal locality and, hence, will make caching useless especially in applications that require real-time certification[10]. However, this is an extreme side of the situation, and many applications and transactions have less strenuous requirements for certificate freshness. For example, CACMAN could be deployed in an environment that accepts a validity period of, say, a couple of days or weeks without CRL.

---

[10] When the client and server clocks are not synchronized, an accuracy of 15 seconds is de facto real-time. Evermore, most financial applications trust 4-hour old signatures[49].

Now, let us consider a more challenging environment that requires both types of certificates[11] (i.e. short and long-lived). In this environment, CRL is needed and CACMAN should distinguish between both types of certificates and allow CACMAN clients to cache long-lived certificates besides handling CRL distribution efficiently.

If we apply OCSP [41], RCA [52], or NOVOMODO [49], we introduce another server to answer queries about revocation status which means a single point of failure has been added to the system. Replicating OCSP will expose the system more and make it more complicated. Another alternative design is to make status checking done by CACMAN CAs themselves. In this case, we utilize the security investment (i.e. threshold based protocol); however, we increase the burden on CA servers and imagine how every queried CA will try to collect at least t agreements to sign replies and this is exactly what we are trying to avoid.

Now, let us consider the fundamental CRL method in which every list contains a field telling when the next issue will be. Since this information is signed by CAs and publicly available, there is no harm in duplicating them to some other nodes by flooding and hoping the list will be delivered to all nodes or by selecting some strategic nodes which could volunteer to carry this burden. The problem arises when a given node missed an update (i.e. the "next update" time elapsed), in this case, if the client does not receive the CRL in the expected time frame, then it should consider its cache as suspicious and purge it. A possible enhancement is to mark the cached partials as dirty without using them, hoping the list would be obtained when the network is connected again by retrieving the missing update from any neighbor or querying any CACMAN server. Delta CRL will have similar situation but with the more challenging problem of probably seeking more than one missing CRL update[12], and it has the advantage of having compact size.

---

[11] For example, short-lived certificates are issued for single onetime login tickets and long-lived ones are issued for authorizing financial transactions that are less than a given amount.

[12] Remember Delta CRL is issued incrementally and the verifier should obtain all issued lists.

Yet another method is to issue counter certificate as implemented in URSA (Section 3.3). The advantage of this method is that it does not put any timing requirements in revoking certificates (i.e. revoke on demand). One problem is to make sure that a counter certificate is propagated to all legitimate nodes. A simple solution, though not bullet proof, is to flood the network whenever a counter certificate is issued. To make the solution even better, every counter certificate issued by the CA should contain an auto increment number. The purpose of that number is to allow any CACMAN client to determine whether it has missed any counter certificate issuance or not. However, this is possible if we assume that any given node is surrounded by one or more legitimate nodes. The client will send, say, a local broadcast message showing the last counter certificate which it has received, and legitimate nodes will tell if the last counter certificate heard by the requester is indeed the latest one.

CACMAN framework has no problem adapting the CRL methods mentioned above, and there are no security threats of caching CRL since it is public information signed by CA servers. The answer to the question "Which CRL method is suitable for MANET?" seems to be application dependant. We think integrating Delta CRL into CA servers is reasonable for most applications while others that need more freshness may request the CA directly.

4.7. Other Issues Associated with the Framework

4.7.1 Key Refreshing between CAs

Key refreshing is an important operation done by CA servers to protect against malicious attackers from learning the secret over a long period. CACMAN nodes do not interfere with that operation. However, there is a problem after refreshing CAs partial-keys: any issued partial certificates are no longer compatible with the shares residing in CACMAN client caches. We emphasize that this incompatibility does not mean those partials cached in CACMAN nodes are obsolete. They are still valid but not compatible with any feature

issued partials[13]. However, a node should somehow have the means to distinguish between old and new shares. One possible way is to make CREPs contain a number that tells in which epoch this partial has been signed. To reduce the overhead of housekeeping, CACMAN node should delete shares belongs to old epoch.

4.7.2 Partial Certificates Compaction and Redundancy Removal

The purpose of this operation, as the name suggests, is to remove redundant shares carried by clients. This process reduces the storage needed to carry certificates since a partially signed certificate is as large as a complete one. Additionally, it saves some bandwidth because a single complete certificate may travel instead of several partial ones, particularly when the threshold is relatively high.

The procedure could be executed periodically[14], and each CACMAN node locally broadcasts a list that contains the missing parts needed to complete its collection; then, surrounding CACMANs will send back missing parts, if available, to the client for reconstruction.

It is possible that some partials will not be combined because they have been originated from a compromised CA or modified by a malicious CACMAN. But as we have mentioned in Section 2.5, detection schemes without exhaustive search will be used to figure out and delete wrong partials.

4.7.3 Distribution of Partially Cached Certificates

One important issue is how to distribute full or partial certificates coherently in the system to reduce the possibility of having the same full or partial certificates residing in adjacent neighbors. We suggest some techniques on how every node decides whether it caches an overheard certificate(s) or hopes that other node(s) do it. Such a decision will

---

[13] Remember that key refreshing does not change the private-key of the service. Otherwise, every node should be informed by such change and this will reduce the usefulness of the service.

[14] It is not necessary that all CACMAN nodes execute the procedure at the same time. Every CACMAN node can decide individually when the next execution to refresh its repository will occur.

affect two things. First, it will affect the possibility for a CACMAN client to find that certificate in the future. Second, it will affect the chance for that client to offer that certificate to other CACMAN clients.

Here, we show some possible techniques on how coherent distribution is possible:

1) *All on the Path Cache:* This is the simplest method, when a CREP message travels through its path, every client in that path will cache the partially signed certificate contained in that reply.

2) *All Can Hear Cache:* In this method, every node that receives a CREP, including nodes that reside in the radio range of the node that forwarded the CREP, will cache the partial certificate. This method will make nodes have similar caches especially in low mobility scenarios. However, in high mobility, and more specifically when each node goes to unrelated directions, this and "All on the Path Cache" can be useful to make an even distribution of the certificates or partials.

3) *Path Flip-flop Cache:* In this case, a node residing in the CREP path inspects the header; if the node order is odd, then it will cache that CREP, including the recipient. This heuristic tries to discriminate certificate distribution because it makes no sense for two adjacent nodes to have almost the same contents. The advantage of this technique is more noticeable in low-mobility scenarios. However, it is not possible for some routing protocols (e.g. AODV) to know if a given node is even or not in a given message path. In that case, a user-defined Boolean field in the CREP header could be used to implement this heuristic by flipping its value in every node the CREP visits and cache only iff the field is set.

4) *Probabilistic Cache:* every node receiving a CREP will cache it based on a predefined probability. This probably should be chosen based on network density (e.g. low probability in dense networks and high probability in low density networks).

Fig. 8 illustrates these alternatives. The "S" node denotes a CACMAN that sent a CREP or a client node that received a CREQ and it has the certificate in question in its local cache. The "R" node denotes the request issuer. Nodes labeled "C" denote clients that will cache that particular CREP and empty nodes are those that will take no action. The arrows denote the CREP path and the gray lines denote the connectivity (e.g. in the radio range of the connected node).

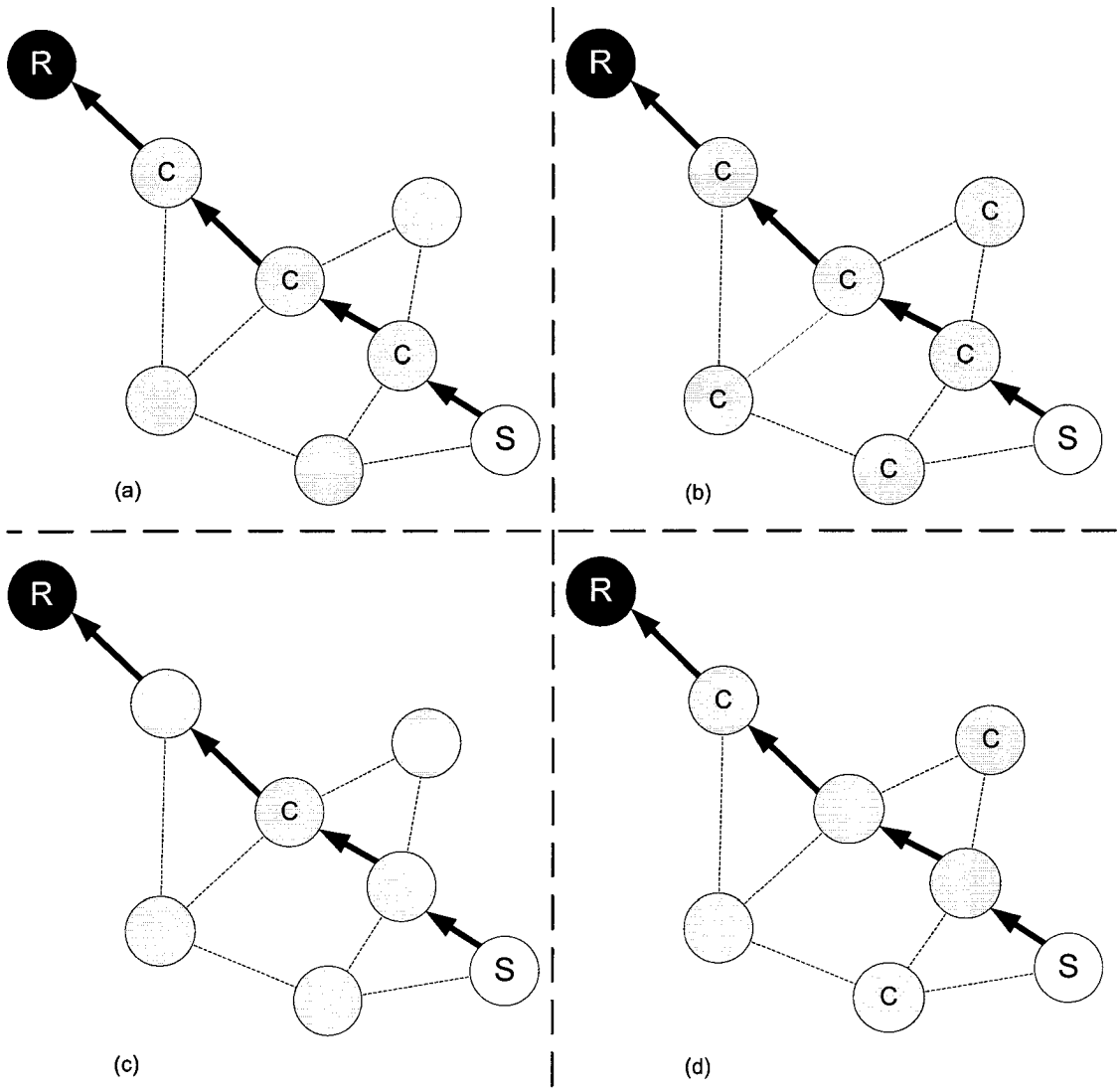


Fig. 8. Cache Heuristics (a) All on the Path Cache (b) All Can Hear Cache (c) Path Flip-flop Cache (d) Probabilistic Cache.

4.7.4 Cache Replacement Policy

When a CACMAN node wants to cache a certificate and it happens that its cache is full, several classic policies could be used to choose the victim, such as LRU or FIFO. Additionally, other possible criteria are based on the completeness of the victim certificate and could be combined to previously mentioned techniques. There are additional possibilities on how to choose the victim:

1) The victim could be the new partial itself since the caching CACMAN client has a complete certificate making the former redundant.
2) The victim partial would be the one corresponding to a certificate that has the minimum number of shares at that client.
3) If the received reply is a complete certificate, the decision is easier; the victim would be those partials that correspond to the complete certificate. If none exist, option 2) could be applied.

4.8. Deployment Scenarios in Hybrid MANET

MANET usually has been seen as an isolated network that does not have a connection to the outside world. This is acceptable in some context and scenarios, but a more generic model is possible, "hybrid MANET." This is a topology that integrates a MANET with a gateway connected to the Internet.

We think of two possible scenarios for deploying CACMAN CA servers in such environments as those illustrated in Fig. 9 and Fig. 10. Both configurations share the following components: a MANET infrastructure connected to an access point which is connected to a fixed infrastructure via a satellite link. The fixed infrastructure in turn is connected, through routers, to a higher level CA server residing on the Internet which may cover other networks. The principal difference between the two configurations is where to deploy CACMAN CA servers. Deploying CACMAN CAs outside the MANET, as in Fig. 9 , will eliminate CA exposure to wireless threats (i.e. more security). Moreover, if we allow some CA to acts as delegates – similar to COCA- a CACMAN

client will be required to contact one CA server to get a complete certificate which eliminates the overhead of caching partials while still caching full certificate. The disadvantage of this design is that the gateway will be a bottleneck on the system, and the inherent propagation delay of satellite connection will reduce response time. Additionally, it will be a single point of failure, and if compromised, CA service in the MANET will be denied.

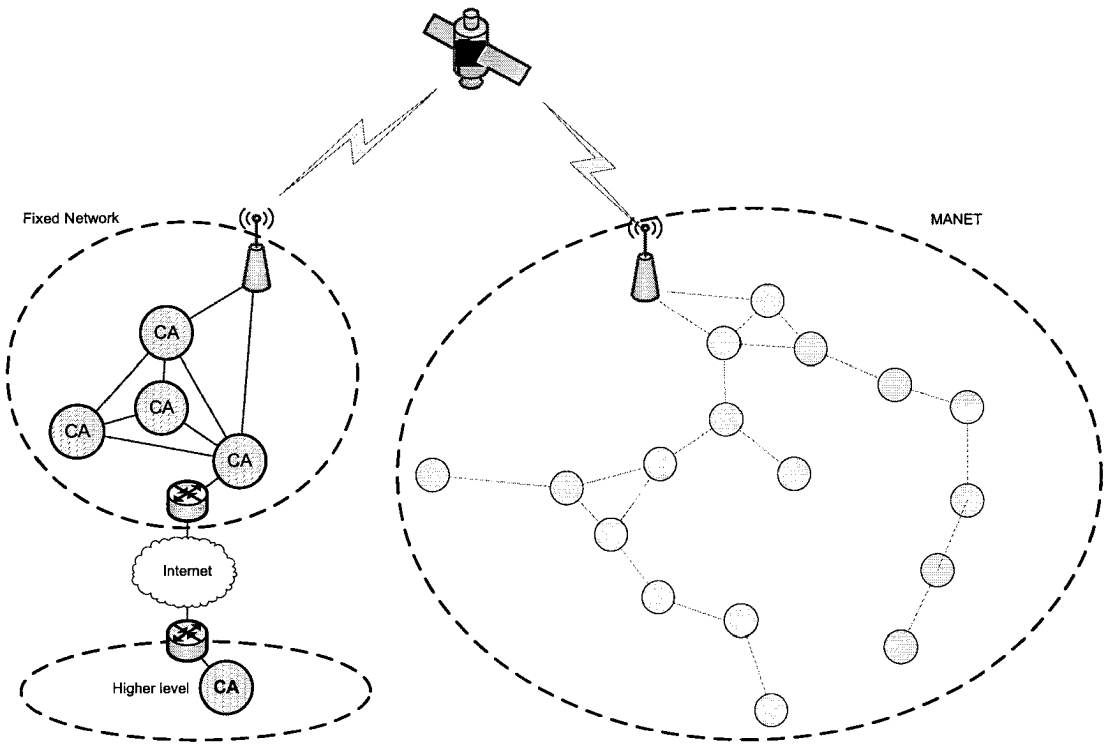


Fig. 9. CACMAN CA servers outside MANET domain.

On the other hand, deploying CA services within MANET will eliminate the frequent access to the access point except when communicating with the higher level CA, which is rarely required, in addition to a fast response time. Deploying CA inside the MANET will require more care regarding CA security since they will be more exposed and probably require a higher threshold for the system, as compared to the former design, to ensure security.

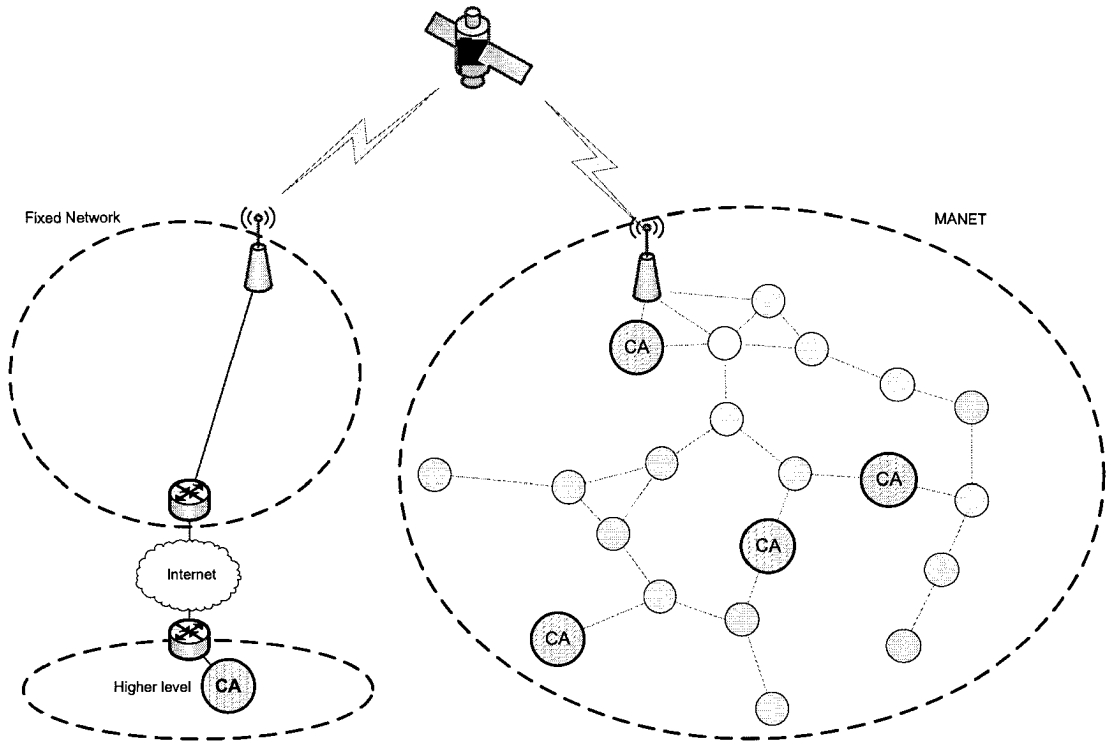


Fig. 10. CACMAN CA servers inside MANET domain.

4.9. Conclusion and Summary

We began this section by showing our motivation to develop CACMAN. We followed by listing the framework assumptions. A detailed description of the basic protocol was given followed by surveying possible communication techniques. Details about various enhancements to our framework were given followed by a discussion on the impact of some CRL schemes on CACMAN. Then we examined other miscellaneous issues such as removing redundancy partials and alterative ways to spread partially cached certificates. Finally, deployment scenarios for hybrid-MANET were discussed. In the next section, we will evaluate CACMAN from performance perspective and show how it reacts to various enhancements.

SECTION 5
SIMULATION AND ANALYSIS

In this section, we put CACMAN to the test. We have implemented and simulated the basic protocol and its variations and enhancements mentioned in Sections 4.3 and 4.5. The purpose of doing this simulation is twofold. First, it provides us a basis for putting CACMAN and related work side by side to get a more objective comparison. Second, it allows us to see how the protocol parameters might behave in practice, hence, allowing us to check their significance to the framework.

At the preliminary stage of this work, we have evaluated CACMAN-Basic using a Monte Carlo simulation that we developed [1]. At that time, this was sufficient to provide us with the preliminary results to continue in this direction, but it was not sufficient to draw a faithful conclusion that could be used to compare it with related work. This is because it is not as realistic as the NS-2 [47], and besides, it does not consider mobility and other parameters. Here, we will show CACMAN simulation results obtained from NS-2 which eliminates the necessity of presenting the outcomes obtained from the Monte Carlo simulation. Additionally, we will restrict our comparison to MOCA protocol for the same reasons as shown in 4.1.

This section is organized as follows: Section 5.1 highlights some details about the simulation environment. Section 5.2 lists the simulation assumptions. The results and discussion are shown in Sections 5.3 and Section 5.4. Section 5.5 is our concluding remarks.

5.1. Simulation Description

In our simulations, we used the most recent build of NS-2[15] to simulate the following scenario similar to the one used in MOCA [70]. The goal of this choice, as mentioned before, is to achieve a fair comparison between the two protocols. Furthermore, we replicated the Flooding scenario described in [70] and [71] to make sure that we have almost identical settings, and it turned out that the differences are negligible.

The scenario used for our simulation follows: 150 mobile nodes were scattered in a 1000 m^2 field, 30 of which are CA servers to the remaining 120 client's nodes. All nodes are moving using the CMU random waypoint model [15]. One hundred nodes will make 10 CREQs during the 600 second simulation time. We have generated nodes mobility scenarios for maximum speeds of 0, 1, 5, 10, and 20 m/sec with pause times of 0 and 10 seconds. Each configuration is replicated at least three times[16] to ensure the accuracy of the results. In the original paper [70], a network of 300 nodes including 50 servers, has been simulated too, but the paper did not show the results due to the similarity to 150 node case. In addition, we used caches capable of holding 75, 150, 225, and 300 shares, and we tested thresholds of 5, 15, 20, and 25. We have used a single cache replacement policy in which it is applied when a CREP arrives and there is no room in the client's local cache. The policy is to delete a share that corresponds to the certificate which has the minimum shares at that client.

For convenience, the simulation settings for the most frequently referenced parameters are summarized in Table II.

---

[15] A short overview of NS-2 can be found in Appendix A .

[16] In our case, replicating the experiment more than three times did not show any significant differences in the results.

TABLE II
SIMULATION PARAMETERS USED IN IMPLEMENTING CACMAN

Parameter		*Value*
Grid size		1000 m^2
Maximum speed		1, 10, and 20 meters/second
Pause time		0 and 10 seconds
Simulation time		600 seconds
Total number of nodes		150
Number of MOCA servers		30
Threshold (t)		5,15,25
Cache sizes		75, 150, and 225
Routing protocol		AODV
Mobility model		Random waypoint
Total number of requests		1000
Cache replacement policy for CACMAN		Certificate with minimum number of shares will be deleted.
Certificate reply policy	CACMAN-Basic	If a full certificate exists, then send it. Otherwise, send one share selected randomly.
	CACMAN-X replies	If a full certificate exists, then send it. Otherwise, send X shares selected randomly if available.
Reply timing	CACMAN-No delay	On cache hit, share(s) are sent immediately.
	CACMAN-Lazy replies	One cache hit, shares(s) are scheduled to be sent on delay periods between (0 and X) where X=10ms, 40ms, 160ms, or 640ms.
	CACMAN-Lazy replies with cancellation option	Similar to CACMAN-Lazy replies but the requester will send cancel message if it has already got sufficient replies.
Cache Configuration	CACMAN-Shuffled	The cache is filled randomly up to its capacity.
	CACMAN-Focused	The cache is filled by shares that belong to its focus group G where $\|G\| = [1..30]$.

5.2. Simulation Assumptions

Here we list our simulation assumptions that govern our results and all interpretations should consider such assumptions.

1) There are many other parameters in the simulation which can be modified, such as antenna type, propagation model, transmission range, MAC layer buffer size, and packet size, etc. We used the default settings provided by the simulator.

2) There are no assumptions on the processing time, and it takes a constant time to do computing operations.

3) The simulation covers the normal scenario and does not consider any threat model or anomaly behaviors done by clients or servers nodes. It is understood that such behavior will have some impact, and we will cover this point in more detail in Section 6.

4) Even if the requester demands a certificate from a principal which resides one hop away, the requester (client) will make a local broadcast regardless of target principal (i.e. node) location.

5) Maintenance and security parts of the protocol such as CRL dissemination or CA Key-refreshing are not implemented because they are out of the scope of what we are measuring.

6) Random waypoint has a recently discovered problem which is described in detail in [69], but we are still using the old method for comparison with MOCA. However, the impact of using the modified Random waypoint model is not significant unless the total simulation time is very long, above 700 sec, and routing information is involved extensively, and this is not the case in our simulation.

7) Whenever a client receives duplicate partials, they are counted once.

8) In the case of a client receiving a full certificate, it will be counted as if the client has received shares$[1..t]$. The impact of this assumption is that when the client receives further redundant shares which belong to $[1..t]$, they will not be counted as mentioned above.

9) There are many variations and values each parameter can take. Our goal is to see the impact on reasonably selected values. Hence, pushing every parameter to its extreme value is not one of our goals. Thus, it is misleading to draw a generalized conclusion about the system with parameters that are far away from what we have used in this section.

10) Certificates are distributed uniformly into CACMAN clients; however, other distributions could enhance CACMAN performance.

5.3. Results and Discussion

In this section, we will analyze the results for every major configuration and the impact of changing the parameters that we care about. We will start by comparing CACMAN-Basic with MOCA.

5.3.1 MOCA and Flooding

Let us first discuss MOCA simulation results obtained from [70]. Fig. 11(a) shows the results of implementing the certification protocol using the pure flooding method [71]. The X axis represents the number of received distinct CREPs, and their corresponding frequency is shown at the Y axis. For example, in the case of running the flooding protocol when the maximum speed is 10 m/s with a pause time of zero, about 50 requests received 25 unique CERPs, and about one or two requests did not get any replies at all. Indeed, flooding is effective; however, it injects a large number of control packets into the network (e.g. imagine if threshold is set to 10 or less). Fig. 11(b) shows the effect of introducing the MOCA protocol. The impact is already reflected as a reduction on the frequency scale as compared to Flooding protocol. The peaks around different β's are caused by MOCA when it uses cached route information to send unicast requests to

MOCA servers, thus eliminating unnecessary flooding of the network. As we can see, the occurrences are still shifted to the right, which indicates a high rate of success, but many redundant replies are also received due to the fact that MOCA floods the network if no sufficient routes to the CA servers exist in the client routing cache table.

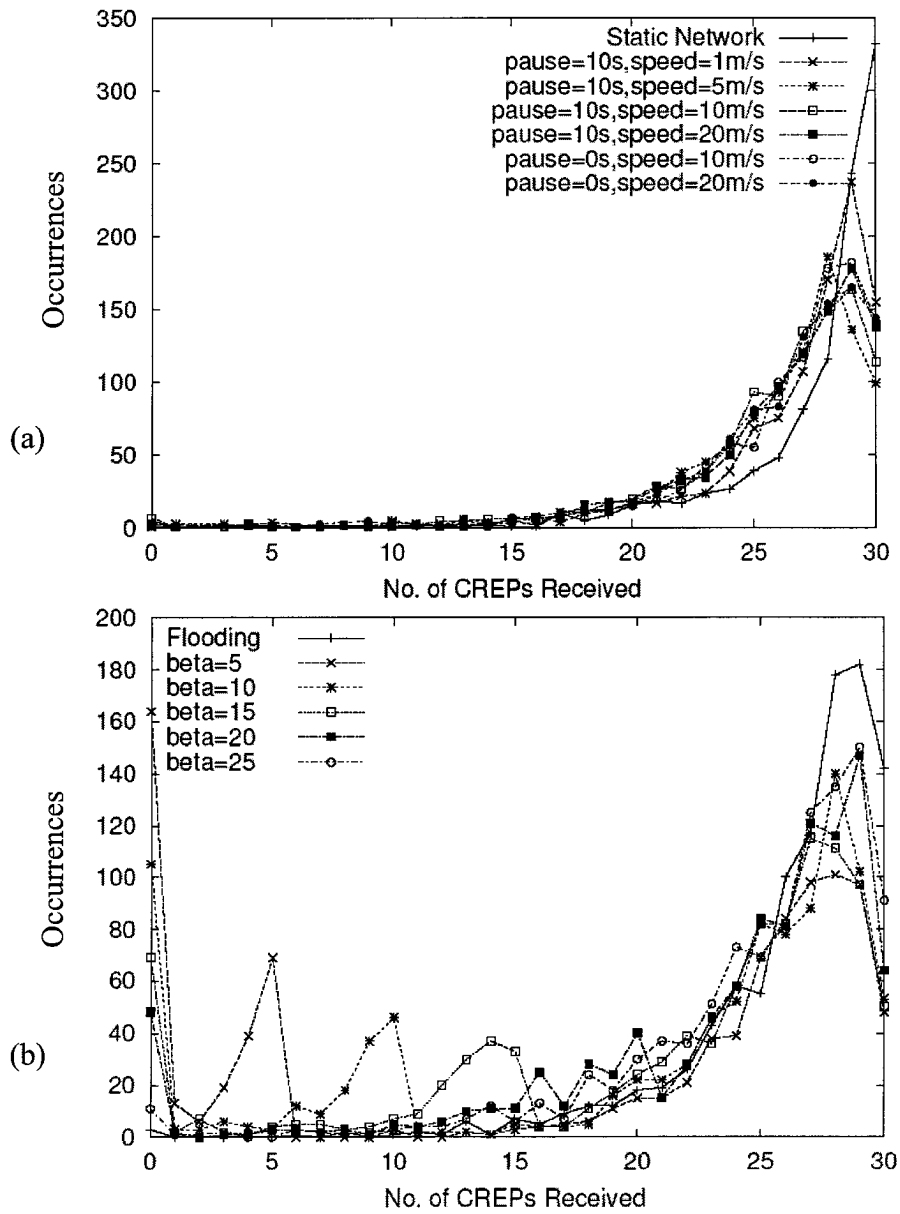


Fig. 11. (a) Flooding based certification protocol. (b) MOCA certification protocol on average speed = 10m/s and 0 pause time.

5.3.2 *CACMAN-Basic and CACMAN-X-Replies*[17]

Fig. 12, Fig. 13, and Fig. 14 show the basic CACMAN protocol when the threshold is set to 5, 15, and 25 respectively. Each of those figures show the effect of mobility and cache size. The legends in these figures give the representations of "number of nodes-pause time-max speed (cache capacity) protocol." In Fig. 12, where $t = 5$, few requests received insufficient replies, and the majority received an overwhelming number of replies. When we increased the threshold to 15, as in Fig. 13, the number of requests which got insufficient replies increased, and the cache size played a more important role. Notice that when the cache capacity increases to 150 and 225, as in Fig. 13 (b and c) respectively, the situation is much better as the number of unsuccessful requests decreases (remember that a request that receives fewer replies than the threshold is considered unsuccessful).

When the threshold is set to 25, the protocol performed badly regardless of the cache size and rarely gets more than a few successful requests out of 1000 requests sent at the simulation time.

From the figures above, we can draw some general conclusions about the performance of CACMAN-Basic with respect to threshold, cache capacity, and mobility:

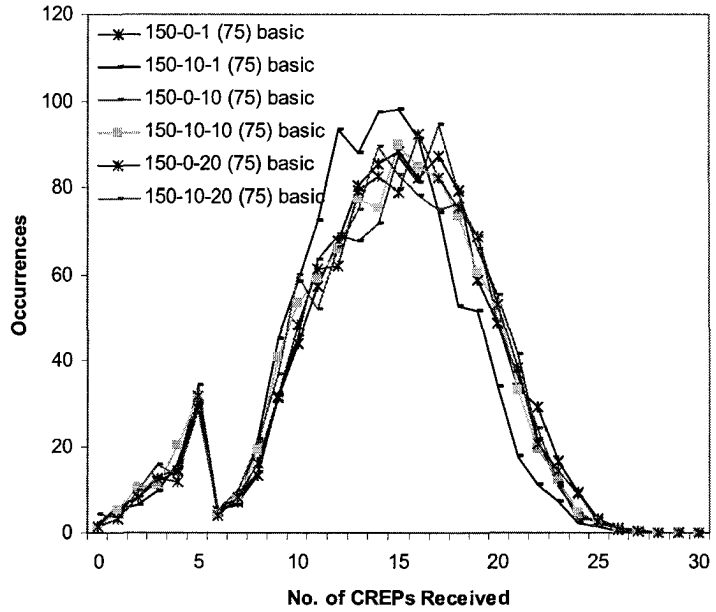
1) The effect of mobility is not that significant regardless of the capacity and threshold and this is true even when other enhancements are used; therefore, we will usually show the results of mobility scenarios of maximum speed of 10m/s and 0 second pause time.

2) We have observed that, when $t = 5$, the cache size of 75 is sufficient, and there is no noticeable benefit when we increase it.

---

[17] The reason that we have distinguished the CACMAN-Basic protocol, which could be seen as 'X-Replies' with X=1, is that we consider the basic protocol as the reference point for many enhancements and comparisons.

(a)



(b)

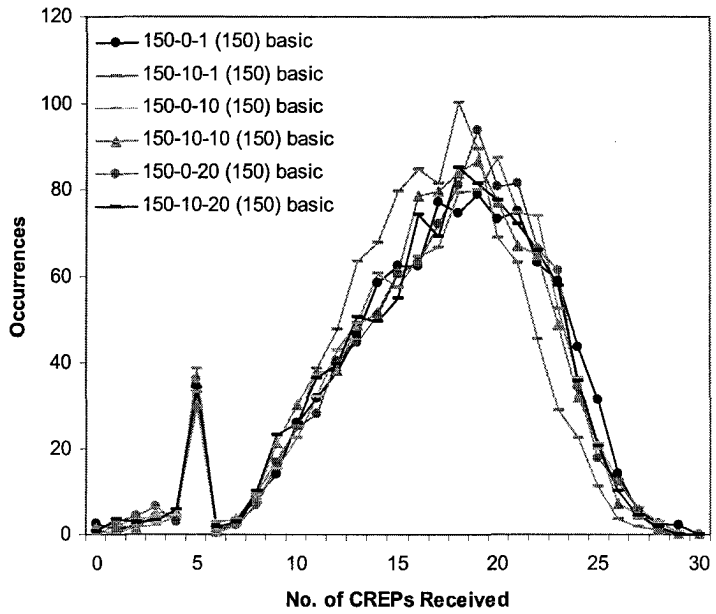
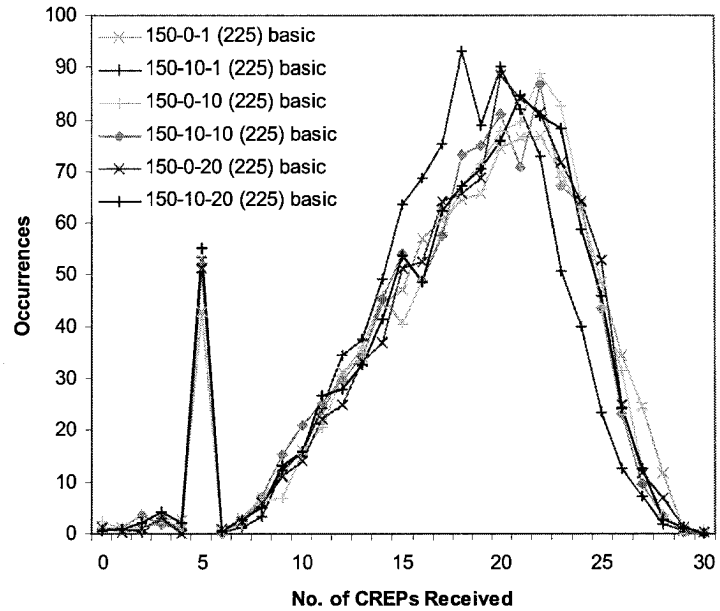


Fig. 12. (t=5). (a-c) Mobility effect with different cache sizes. (d) Cache capacity effect.

3) When $t = 25$, CACMAN-Basic is not capable of satisfactory performance, and the maximum cache size we used was not enough to get significant results.

(c)



(d)

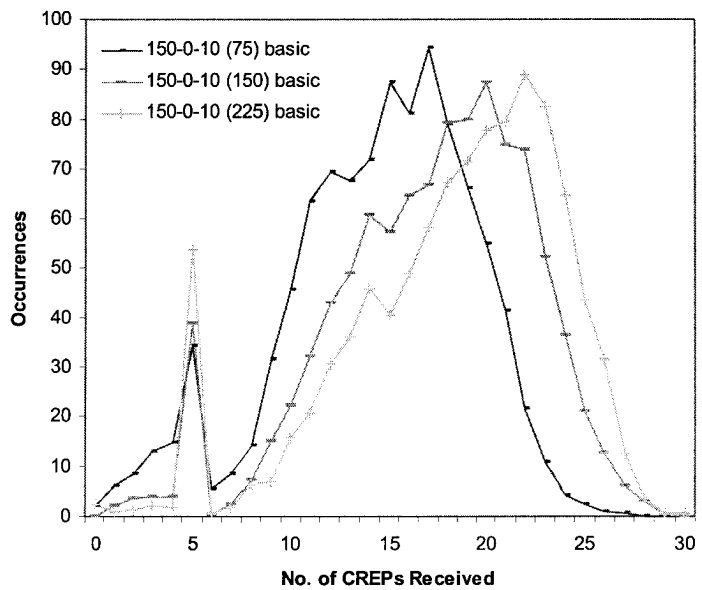


Fig. 12. Continued.

4) At $t = 15$, CACMAN-Basic performance, as expected, was between $t = 5$ and $t = 25$. Thus, it seems that gained performance is worth the increase in cache

size, particularity when it increases from 75 to 150 shares. Finally, the peaks around the thresholds (very noticeable around $t = 5$, less at $t = 15$, and none at $t = 25$) are due to local cache hits. In that case, no request will be sent to the network and this is why the frequency at corresponding t is low, specifically when $t = 15$ or 25.

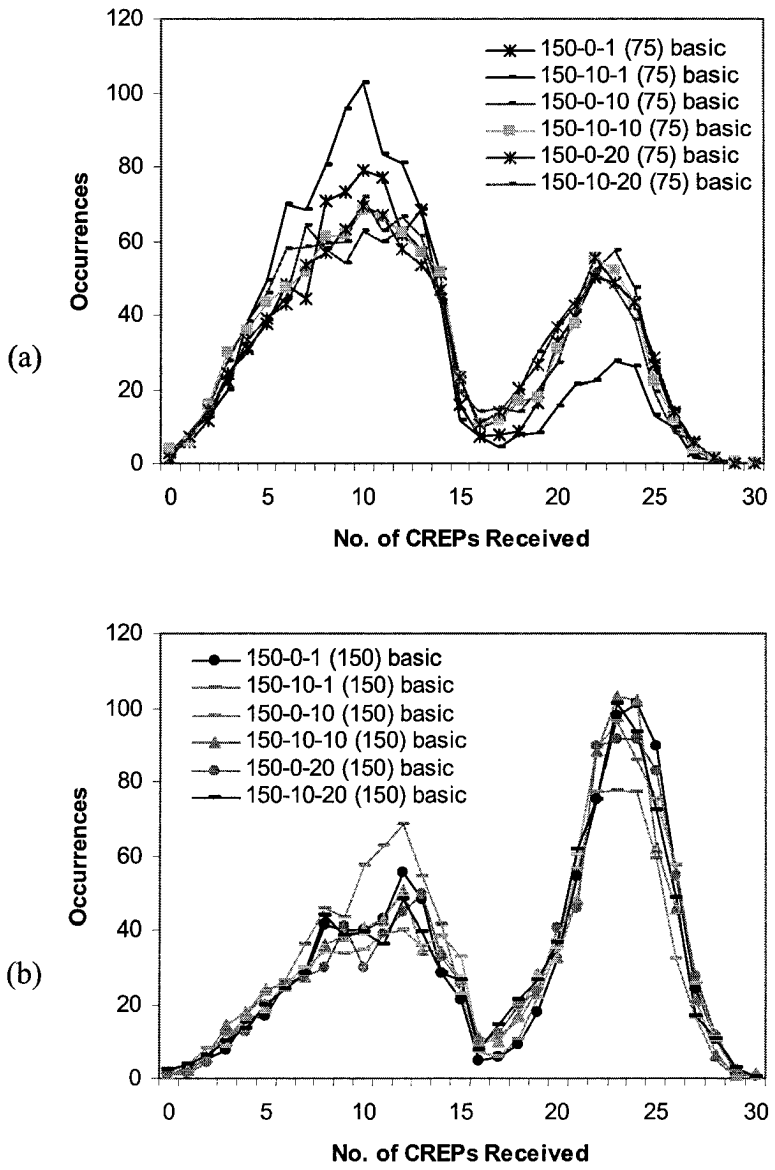
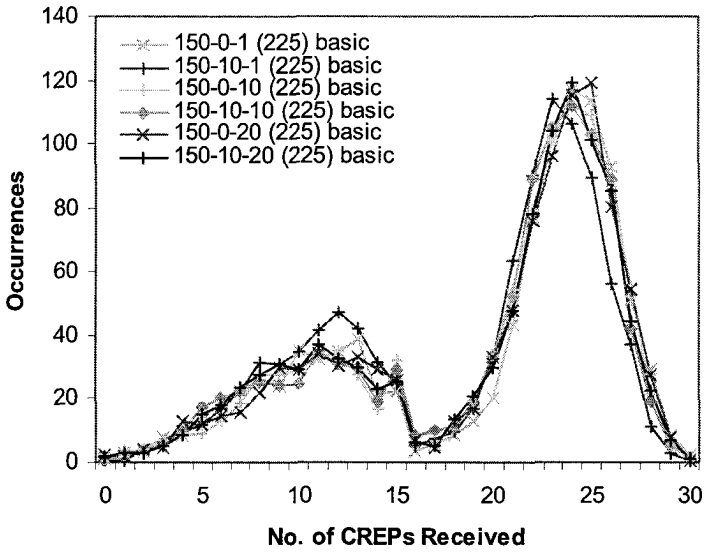


Fig. 13. (t=15). (a-c) Mobility effect with different cache sizes. (d) Cache capacity effect.

(c)



(d)

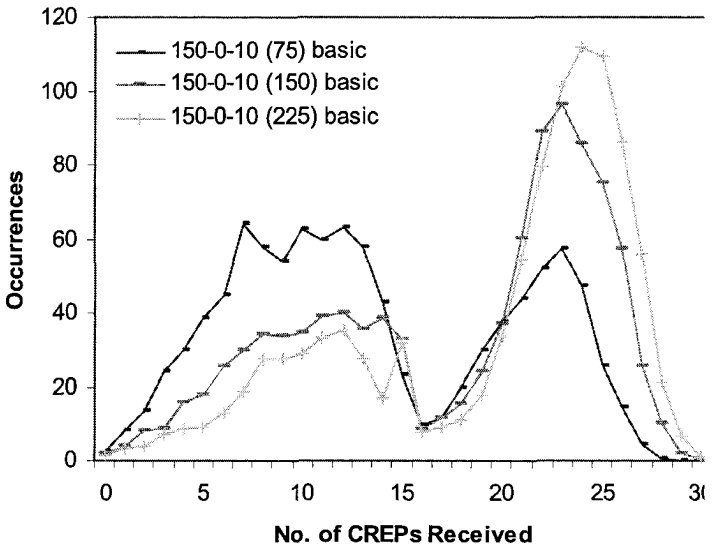
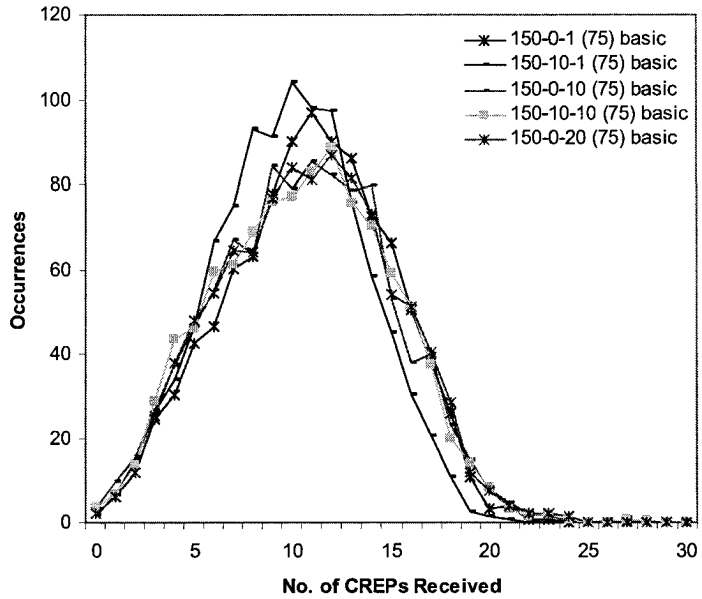


Fig. 13. Continued.

(a)



(b)

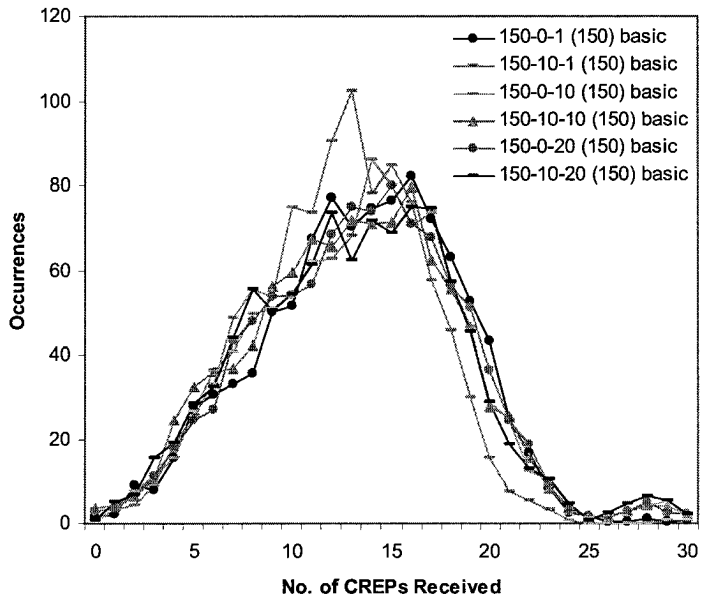
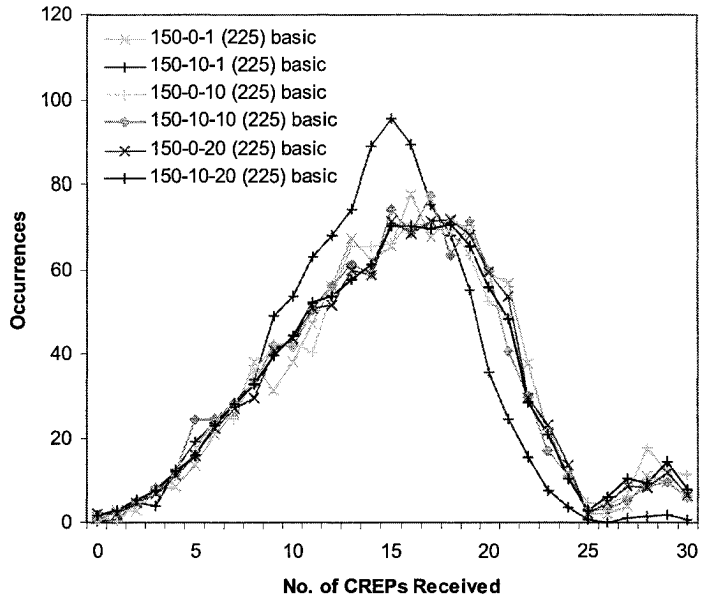


Fig. 14. (t=25). (a-c) Mobility effect with different cache sizes. (d) Cache capacity effect.

(c)



(d)

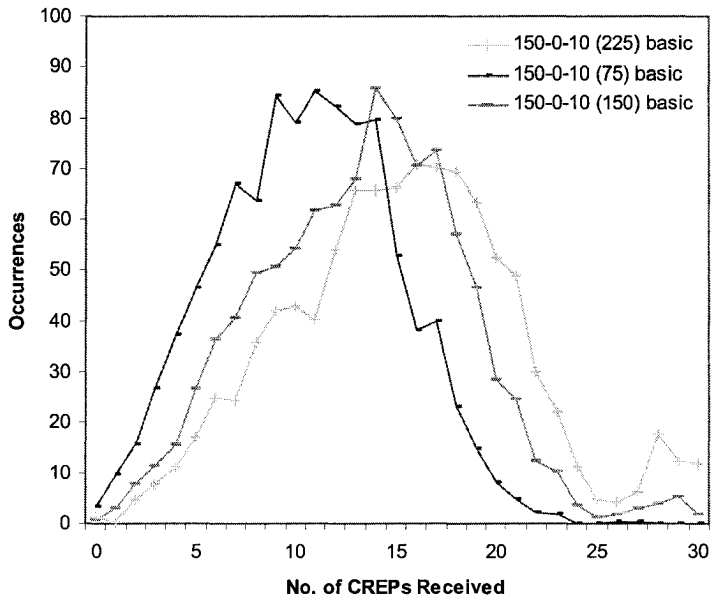


Fig. 14 Continued.

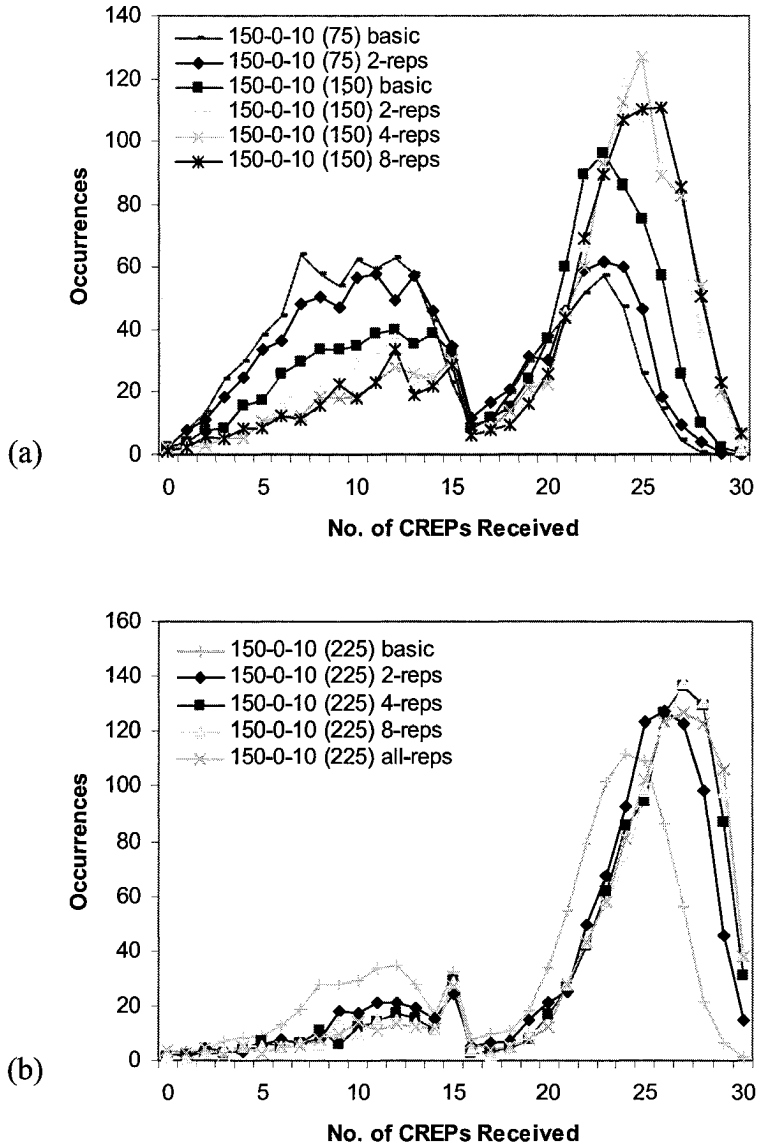


Fig. 15. The effect of applying X-replies optimization. (a) sizes 75 and 150 (b) size 225. t=15.

Now, let us examine the performance gained when we use CACMAN-X-Replies variation. Fig. 15 and Fig. 16 show the impact of increasing the number of replies when a CACMAN client receives a request that has more than one partial in the cache. As the size of the cache gets bigger and the number of allowed replies increases, more successful requests are getting through. Indeed, CACMAN-X-Replies are very useful even when $t = 25$ as shown in Fig. 15. This tells us that the needed partial probably is not

the first one found in the clients' cache, and the more we increase the number of replies, the higher hit ratio we will obtain.

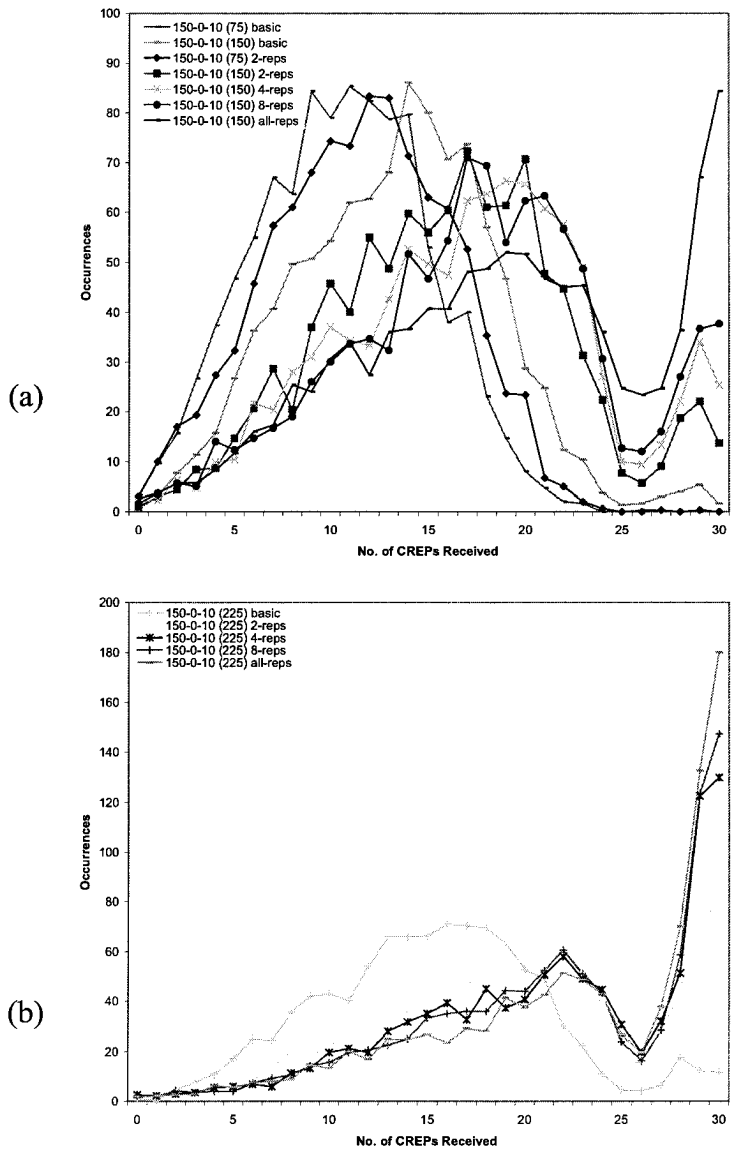


Fig. 16. The effect of applying X-replies optimization. (a) sizes 75 and 150. (b) size 225. t=25.

5.3.3 Comparing Flooding, MOCA, CACMAN-Basic, and CACMAN-X-Replies

Figures shown in Sections 5.3.1 and 5.3.2, gives us a good visualization of how the numbers of CREPs are distributed and also gives an indication of the impact of a given parameter. However, it is difficult to get quantitative measures to answer questions like "How *much* performance do we gain if we change that parameter?" To answer these types of questions, we need to have comparison criteria such that we can see which configuration is better in a given comparison factor. However, we will use the above mentioned way of plotting results accompanied with a supporting numerical value to get a more accurate picture. The performance criteria that we use in this dissertation are the following:

1) *Packet Overhead:* This is a very commonly used criteria to compare networking protocols, and it is interpreted in this context as the sum of data and control packets injected in a given simulation run.

2) *Success Ratio:* In this context, this is defined as the number of successful requests (ones that received at least t replies) over the total number of requests injected during the simulation time.

3) *Overhead per Successful Request:* This is defined as the ratio of packet overhead to the number of *successful* requests. In other words, on average, what is the packet overhead to send a *successful* request?

4) *Adjusted Overhead per Request to Guarantee 100% Success Ratio:* This measures the cost per request to guarantee 100% reliability of the system. Detailed discussion of this criterion will be shown at the end of this section.

In general, to make these measurements useful, compared systems should receive identical or similar workload, and as expected, a system with fewer packets overhead performs better. However, guaranteeing similar workload is not sufficient to draw a final conclusion because of the existence of important factors (i.e. success ratio). To clarify this point, consider system X with packet overhead of 1000 and success ratio of 90% and system Y with packet overhead of 100 and success ratio of 70%. As we can see, it is difficult to judge which system is better if we are considering both factors. To overcome

the above dilemma, either we compare the systems by one factor at a time, using a formula that covers the measurements of interest, or we agree on a common averaging formula and assign some weights to each factor[18]. In this thesis, we will use the first two measures since the third one is subjective. In this section, we will evaluate the systems by showing the packet overhead of the tested configurations and their success ratio. The third and forth measurements will be shown at the end of this section.

TABLE III
PACKET OVERHEAD FOR SOME SELECTED CONFIGURATIONS

Model	*Average number of packets*	*Ratio to flooding*
Flooding	197601	100%
MOCA Closest unicast $\beta = 5$	137666	69.7%
MOCA Closest unicast $\beta = 10$	159039	80.5%
MOCA Closest unicast $\beta = 15$	170375	86.2%
MOCA Closest unicast $\beta = 20$	177364	89.8%
MOCA Closest unicast $\beta = 25$	187063	94.7%
CACMAN basic,75 cache, $t = 5$	16516	8.4%
CACMAN basic,75 cache, $t = 15$	14028	7.1%
CACMAN basic,150 cache, $t = 15$	19121	9.7%
CACMAN basic,225 cache, $t = 15$	22757	11.5%
CACMAN basic,225 cache, $t = 25$	21911	11.1%
CACMAN 2-reps,150 cache, $t = 15$	25564	12.9%
CACMAN 2-reps,225 cache, $t = 15$	33569	17.0%
CACMAN 4-reps,150 cache, $t = 15$	27838	14.1%
CACMAN 4-reps,225 cache, $t = 15$	39448	20.0%
CACMAN 8-reps,150 cache, $t = 15$	28429	14.4%
CACMAN 8-reps,225 cache, $t = 15$	40199	20.3%
CACMAN all-reps,150 cache, $t = 15$	28992	14.7%
CACMAN all-reps,225 cache, $t = 15$	40329	20.4%
CACMAN all-reps,300 cache, $t = 15$	51581	26.1%

[18] The problem of evaluation systems performance has been discussed in depth in [38].

Table III shows the ratio of various protocols' packet overhead to flooding overhead, and for example, MOCA saves about 30% when $\beta = 5$ compared to flooding, while CACMAN saves 92% compared to flooding when $t = 5$ (Remember $\beta = \alpha + t$, but we assume $\alpha = 0$ to simplify the comparison. However, the reader can vary α, and, for example, can compare MOCA with $\beta = 10$ to CACMAN with $t = 5$ by setting $\alpha = 5$). Furthermore, the overhead is also affected proportionally, in the case of CACMAN, by the cache size and the number of replies. CACMAN has a very low overhead compared to both MOCA and pure flooding because CACMAN never allows a request to propagate more than one hop.

If we switch the focus to the success ratio as a measure of performance, Fig. 17 shows the success ratio for various scenarios[19]. We used the same technique used in [70] to plot the CACMAN-Basic and CACMAN-X-Replies success ratios. The figures represent the fraction of successfully arrived CREPs as function of time. According to these figures, flooding is indeed the most effective way to obtain a high success ratio, but at the price of higher packet overhead. CACMAN-Basic performs better, 92% to 99% depending on cache size, compared to MOCA, 91%, when $t = 5$. The cache size has a noticeable effect in CACMAN-Basic, when $t = 15$, (the success ratio was 30%, 60%, and 70% for cache sizes 75, 150, and 225, respectively), but this is not close, as expected, to Flooding (95% at $t = 15$) but close to MOCA (77% at $t = 15$). However, the use of X-replies optimization tightens the gap between cache sizes and gives better results (8-replies with size cache size of 150 achieved 90% success ratio) compared to MOCA, which is close to Flooding.

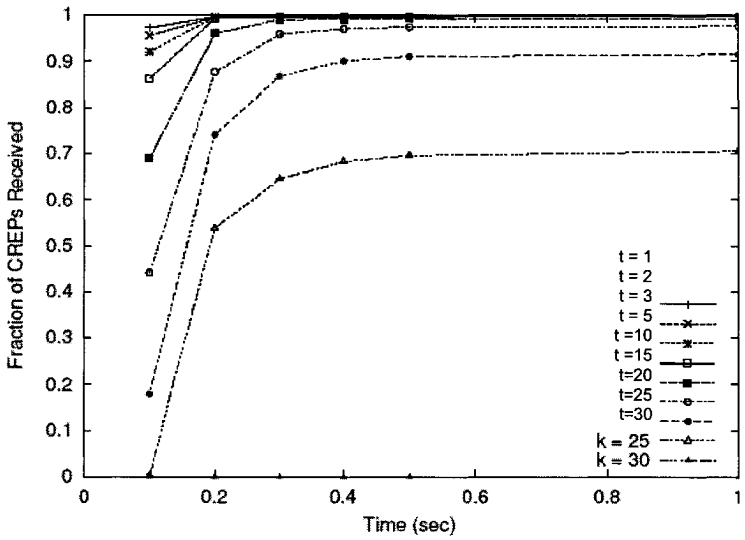
These figures show some other important results. For example, observing the difference in time scale, CACMAN stabilizes (stops increasing) much faster than flooding and MOCA by as much as 16 times (e.g. CACMAN-Basic t5(75) in Fig. 17(c) stabilizes in 0.03 sec as compared to Flooding, in Fig. 17(a), where t=5 which stabilizes

---

[19] The threshold t is denoted by k in [67].

at 0.5sec). Such high performance is due to the fact that some partials are available at the local cache of the requester. Besides CACMAN initial requests never propagate more than one hop. Moreover, CACMAN receives CREPs much faster than MOCA and Flooding, which is another benefit of CACMAN. The previous figure is useful to choose the appropriate timeout value when a CREQ is sent. For example, for CACMAN-Basic with $t = 5$, a reasonable timeout is 0.06 sec and 0.08 sec when $t = 15$.

(a)



(b)

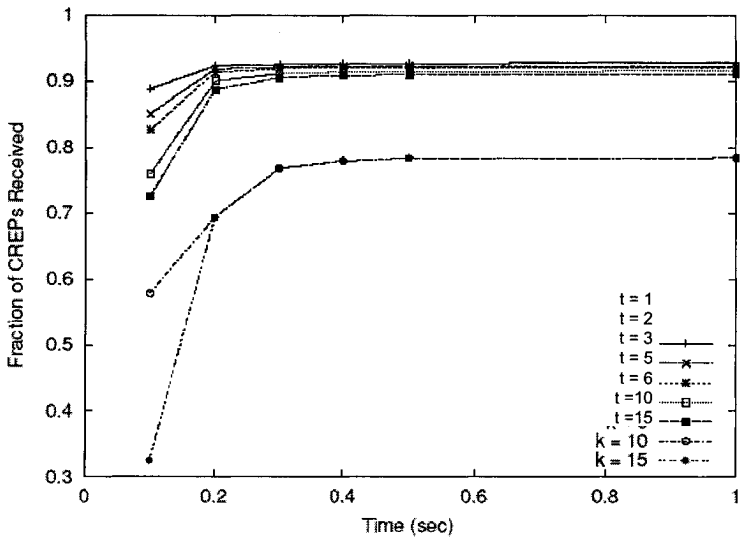
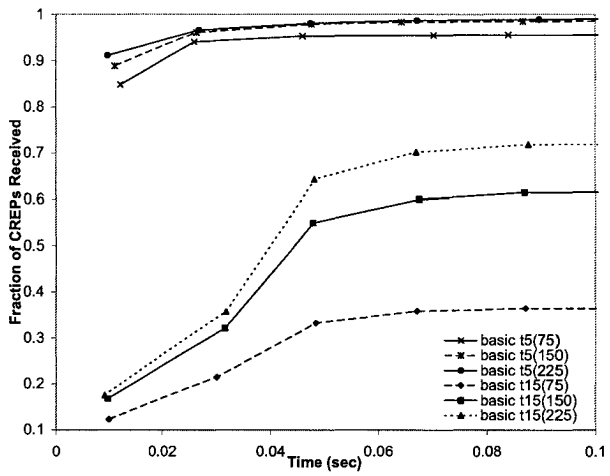


Fig. 17. Success Ratio with (a) Flooding, (b) MOCA Closest-Unicast, (c) CACMAN-Basic, (d) CACMAN X-replies.

(c)



(d)

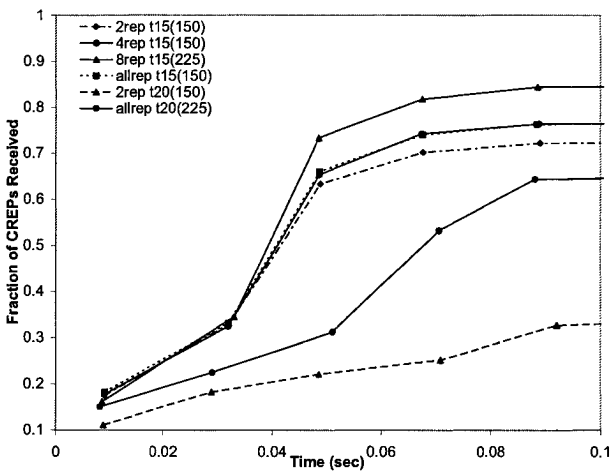
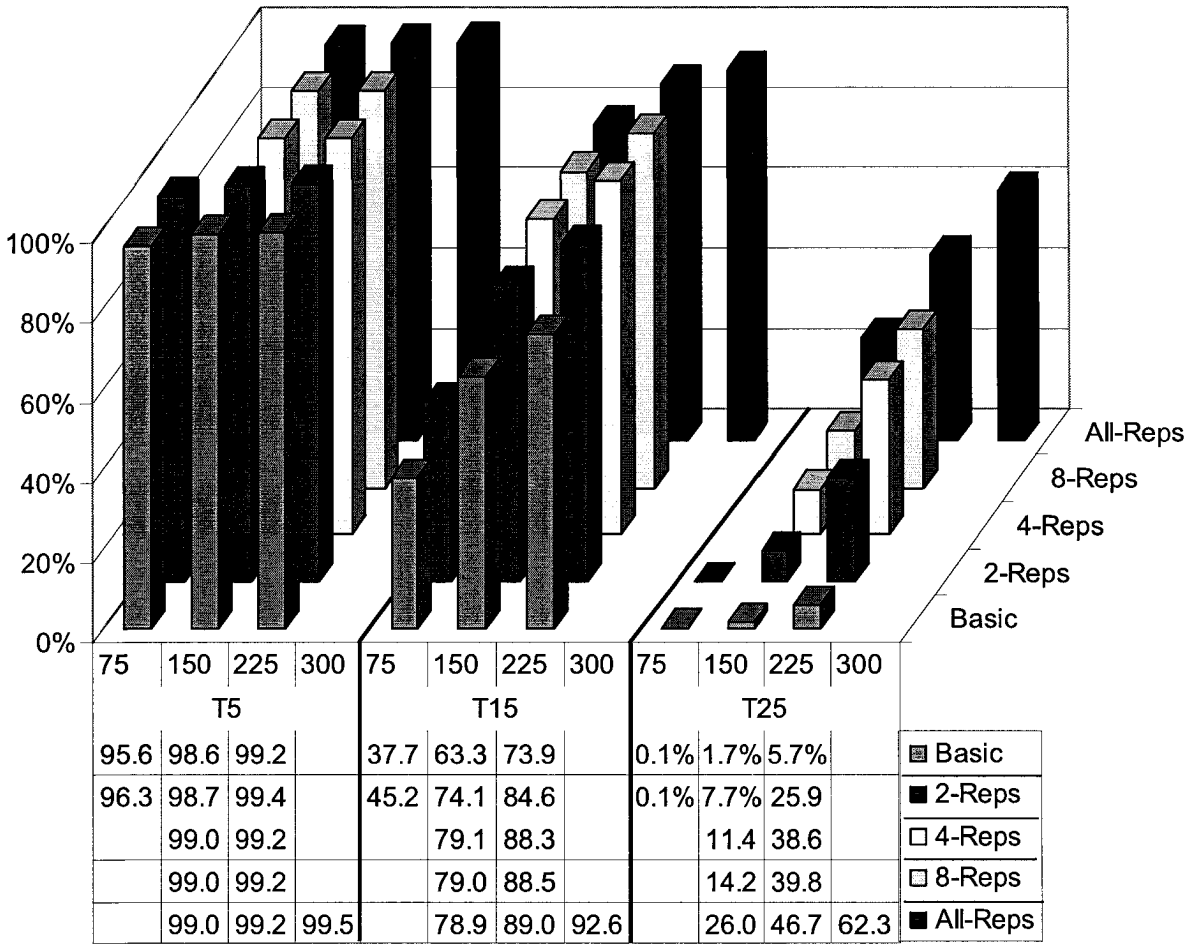


Fig. 17. Continued.

Finally, we would like conclude this analysis by summarizing the success ratio of all the parameters of interest for CACMAN-Basic and CACMAN-X-Replies as shown in Fig. 18. It is clear that the size increase (from 75 to 300) of the cache and the number of replies (from 2 to all found partials) can boost the success ratio specifically when t is high (e.g. 25).



	75	150	225	300	75	150	225	300	75	150	225	300	
	T5				T15				T25				
Basic	95.6	98.6	99.2		37.7	63.3	73.9		0.1%	1.7%	5.7%		
2-Reps	96.3	98.7	99.4		45.2	74.1	84.6		0.1%	7.7%	25.9		
4-Reps		99.0	99.2			79.1	88.3			11.4	38.6		
8-Reps		99.0	99.2			79.0	88.5			14.2	39.8		
All-Reps		99.0	99.2	99.5		78.9	89.0	92.6		26.0	46.7	62.3	

Fig. 18. Success ratio for various scenarios (Max speed = 10m/sec, Pause time=0). Empty boxes mean the results has not been reported

5.3.4 The Effect of Removing CA Servers on CACMAN Performance

This section is to simulate the performance of CACMAN when some CA servers are down because of a sudden crash or a maintenance necessity. The aim of such a simulation is to see how the quality of the service (i.e. success ratio) is affected by the loss of some CA servers. We simulated this behavior by randomly choosing some CAs and turning them off for the entire simulation period.

Fig. 19 shows the impact of losing 10 and 20 CAs ($t = 15$), a loss of one third and two thirds of the total number of CAs respectively as compared to when all CAs are functioning. We used CACMAN-Basic and CACMAN-All-Replies clients with different

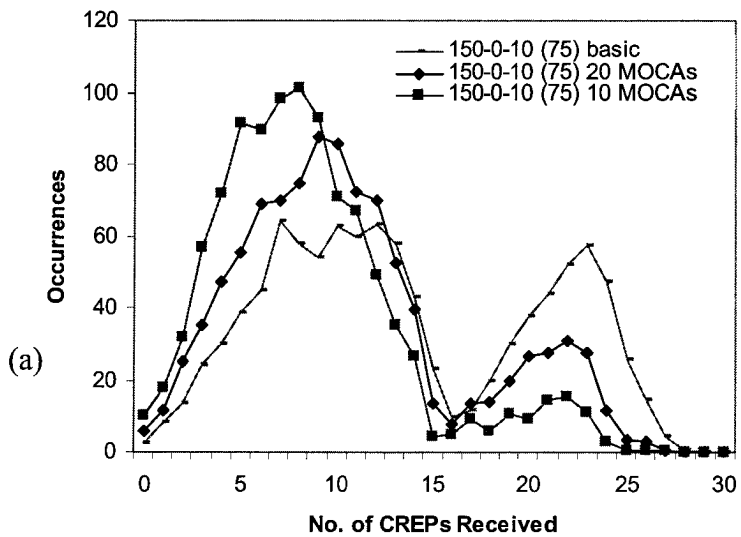
cache sizes to see the tolerance of different configurations to such losses. The figures show, clearly, that a large cache size or the use of All-Reps variance plays a major role in making the service transparent (Fig. 19(e)). We mapped Fig. 19 to Table IV and Table V for a clearer analysis.

TABLE IV
SUCCESS RATIO WHEN SOME CA SERVERS CRASH (CACMAN-BASIC)

# CAs	*150 cache*	*225 cache*
30 (default)	63.3%	73.9%
20	51.3%	65.8%
10	40.0%	60.7%

TABLE V
SUCCESS RATIO WHEN SOME CA SERVERS CRASH (CACMAN-ALL-REPLIES)

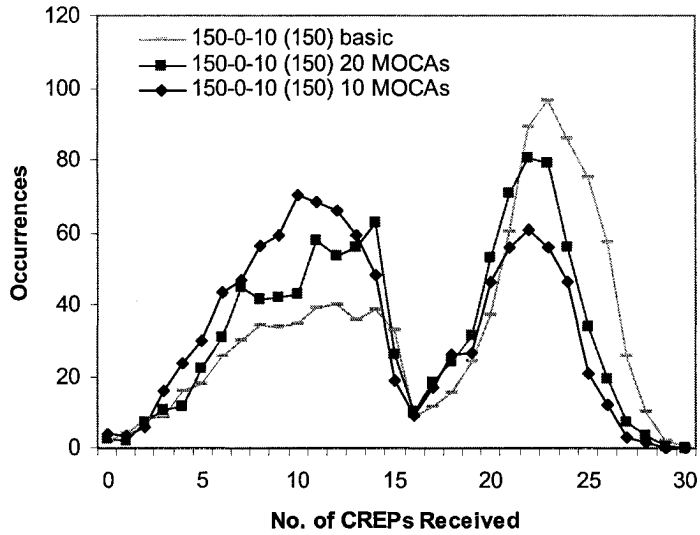
#CAs	*150 cache*	*225 cache*
30 (default)	78.9%	89.0%
20	74.4%	86.2%
10	68.9%	84.5%



(a)

Fig. 19. Visualization of the impact of CA crashes.(a) CACMAN-Basic with cache size 75, (b) with cache size 150, (c) with cache size 225. (d) CACMAN-All-Replies with cache size 150, (e) cache size 225. All with t=15 and max speed of 10 m/sec.

(b)



(c)

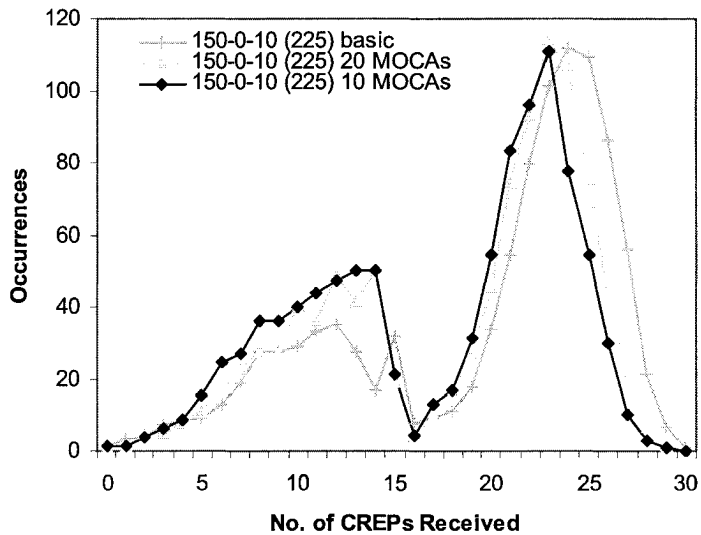
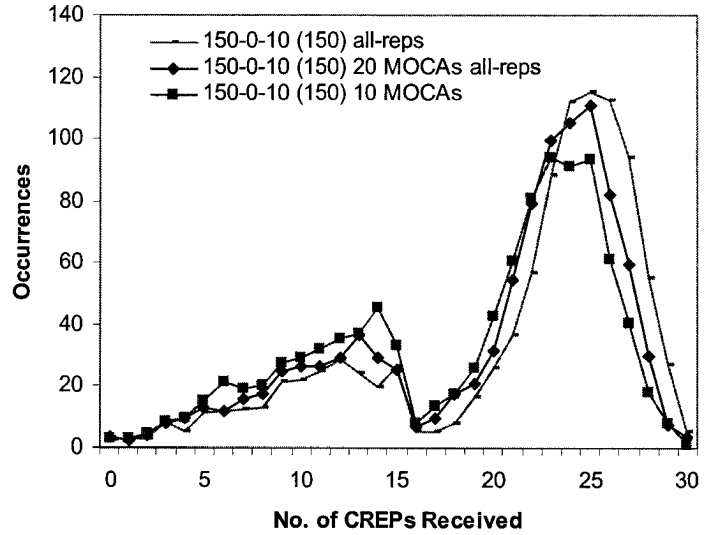


Fig. 19. Continued.

(d)



(e)

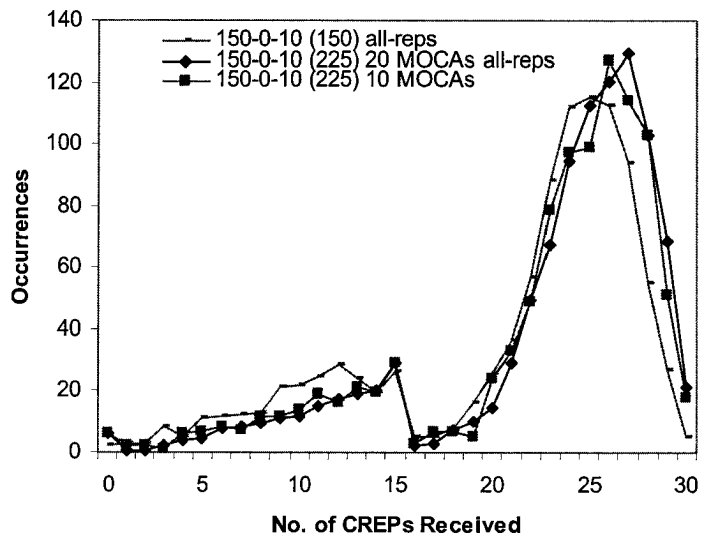


Fig. 19. Continued.

5.3.5 Cache-Focus Optimization

In Section 4.5.3, we described the rationale behind Cache-Focus (CF) optimization. Here, we will show the impact of implementing Cache-Focus into the protocol performance. We have applied this enhancement on CACMAN-Basic and CACMAN-All-Reps with

the values of 1, 2, 4, and 8. This means in each simulation run, every client will focus on caching partials issued by one CA (i.e. CF=1), any two CAs (i.e. CF=2), and so on.

The results show an initial improvement when CF=1, but it starts to be less effective as CF increases, and when CF=8, the hit ratio is almost equal to CACMAN-Basic as shown in Table VI. The reason why a small CF value is beneficial is because it increases the chances of finding a partial for the target by allowing only shares that belong to the chosen CF to be exist at any client. For example, in the case when the cache size is 150, a partial is always found (remember we have a total of 150 participants), and the hope now is to find *t* neighbors with different cache-focus choices to guarantee distinct replies. On the other hand, CACMAN-All-Reps eliminates the benefits of this optimization by sending many replies and hoping that some of them will match the needed request as shown in Table VII.

TABLE VI
CACHE-FOCUS FOR CACMAN-BASIC

Protocol	75 cache	150 cache
Basic	37.7%	63.3%
1 CF	55.3%	83.1%
2 CF	53.5%	82.3%
4 CF	53.5%	81.9%
8 CF	34.3%	59.6%

TABLE VII
CACHE-FOCUS FOR CACMAN-ALL-REPS

Protocol	75 cache	150 cache
All reps	51.1%	78.9%
1 CF	55.0%	80.0%
2 CF	53.1%	81.2%
4 CF	51.7%	77.7%
8 CF	47.4%	77.9%

5.3.6 Lazy-Replies with Cancellation

In Section 4.5.2, we mentioned that the goal of the lazy replies technique is to reduce the amount of dropped packets due to the packet-storm-effect (i.e. packets arrive to a single node from many sources at the same time). We implemented this enhancement by

delaying CREPs by 10, 40, 160, 640 milliseconds. The simulation shows that there is no significant gain in the performance because the CACMAN's low packet overhead reduces the impact from such situation. However, even if CACMAN has immunity from the packet-storm-effect, as compared to MOCA and Flooding, there is still a good reason to implement the other variation, i.e. Lazy-Replies with Cancellation, to reduce the amount of redundant replies by allowing the client to send a "Cancel" message once it has the sufficient number of CREPs. Fig. 20 illustrates the effect of implementing various delay values. The goal of this enhancement is to group the number of CREPs as close as possible to the threshold. This will not affect the success ratio, but it positively reduces the unnecessary CREPs. Fig. 20(a) suggests that there is no significant difference between a 10 and 40ms delay; however, lengthening the interval to 160 and 640 milliseconds pushes the number of replies close to the threshold (here, $t = 15$) . The figure also suggests that there is no significant difference, again, between 160 and 640 ms delay. However, CACMAN-All-Reps, Fig. 20(b), shows the same results when 10 and 40ms values are examined despite a noticeable gap between 160 and 640 ms delay. Remember that the cancellation message will not be sent until the client which has sent the CREQ gets sufficient CREPs. In the case of CACMAN-All-Reps, if a CACMAN client has more than one CREP, each one will be scheduled for transmission, and a cancellation message will purge CREPs which have not been sent yet. This explains why CACMAN-All-Reps benefited more from this enhancement as compared to CACMAN-Basic.

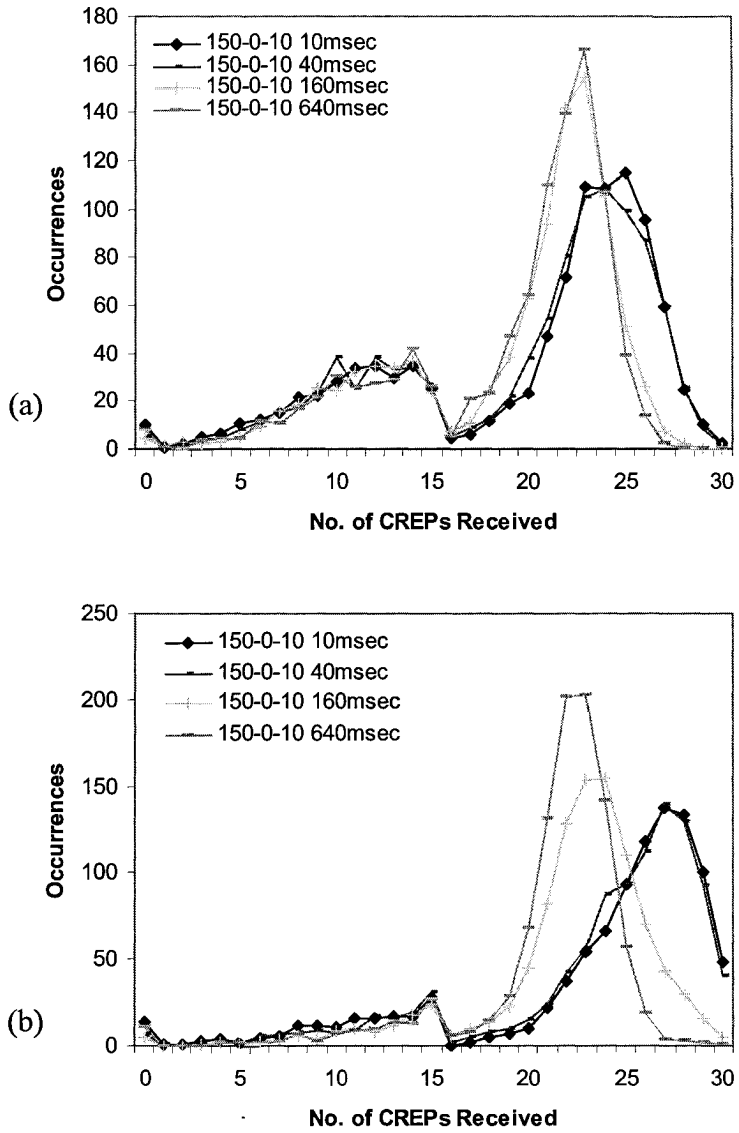


Fig. 20. Visualization of different delay values (t=15) (a) CACMAN-Basic (b) CACMAN-All-Reps.

Fig. 21 shows the packet overhead for both scenarios. For example, in the case of CACMAN-All-Reps with lazy reply interval 640 ms, there was 55% less packet overhead compared to 10 ms delay.

Finally, we should mention that the amount of reduction of packet overhead is also a function of the threshold. Intuitively, when $t = 25$, there is a little reduction since the amount of redundancy is less compared to $t = 5$.

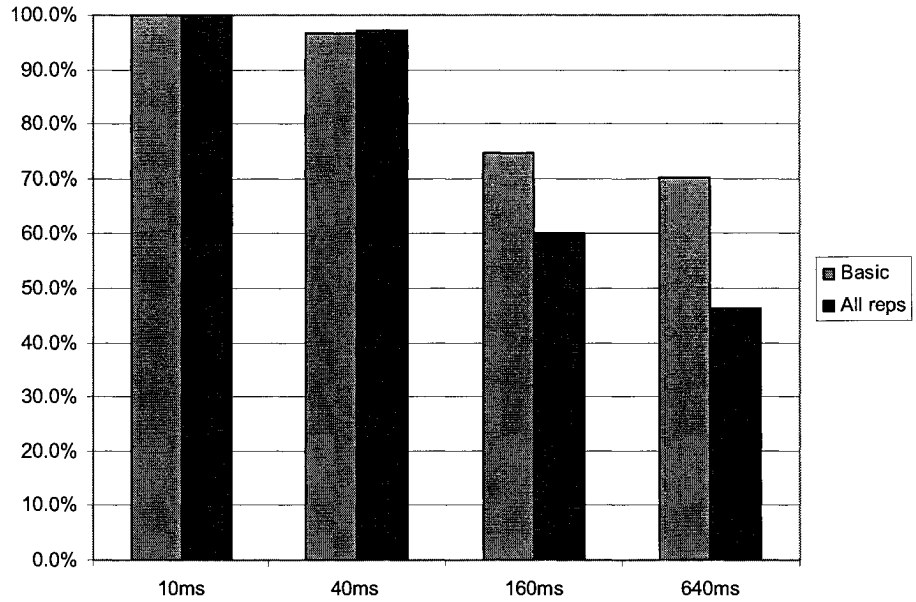


Fig. 21. Packet overhead for different maximum-delay values (normalized to 10ms delay).

5.4. Single Measure

In all of our previous discussions, measurements of Flooding, MOCA, and CACMAN performance were based primarily on the success ratio and packet overhead. We have shown that it is desirable to increase the former and decrease the latter. We will conclude this section by showing that both factors can be merged to a single one as described in Section 5.3.3. Table VIII lists packet overhead per successful request and the adjusted packet overhead to guarantee the 100% success ratio for a request assuming a client will use flooding if the first attempt was not successful. That single measurement is calculated by the following formula:

$$AO = (\text{hit ratio} \times POSR) + ((1 - \text{hit ratio}) \times FOR)$$

AO stands for "Adjusted Overhead to ensure 100% reliability" and *POSR* is "Packet Overhead per Successful Request" and *FOR* is "Flooding overhead/request."

TABLE VIII
POSR AND AO VALUES FOR VARIOUS CONFIGURATIONS

Protocol	*POSR*	*AO*
Flooding	199.597	199.597
MOCA ($t=15$)	218.430	214.290
CACMAN-Basic $t=15$, cache size 75	37.210	138.380
CACMAN-Basic $t=15$, cache size 150	30.207	92.373
CACMAN-Basic $t=15$, cache size 150, Lazy 40ms	30.538	74.662
CACMAN-Basic $t=15$, cache size 150, Lazy 640ms	21.486	63.698
CACMAN-Basic $t=15$, cache size 150, 1 FC	40.764	67.607
CACMAN-Basic $t=15$, cache size 150, 2 FC	41.015	69.084
CACMAN-Basic $t=15$, cache size 150, 4 FC	41.340	69.984
CACMAN-Basic $t=15$, cache size 225	30.794	74.859
CACMAN-2 Reps $t=15$, cache size 150	34.499	77.260
CACMAN-2 Reps $t=15$, cache size 225	39.680	64.307
CACMAN-4 Reps $t=15$, cache size 150	35.193	69.554
CACMAN-4 Reps $t=15$, cache size 225	44.675	62.801
CACMAN-8 Reps $t=15$, cache size 150	35.941	70.145
CACMAN-8 Reps $t=15$, cache size 225	45.423	63.153
CACMAN-All Reps $t=15$, cache size 150	36.745	71.107
CACMAN-All-Reps $t=15$, cache size 150, Lazy 40ms	44.145	61.711
CACMAN-All-Reps $t=15$, cache size 150, Lazy 640ms	20.372	35.427
CACMAN-All-Reps $t=15$, cache size 150, 1 FC	41.542	73.153
CACMAN-All-Reps $t=15$, cache size 150, 2 FC	41.665	71.356
CACMAN-All-Reps $t=15$, cache size 150, 4 FC	43.061	79.065
CACMAN-All Reps $t=15$, cache size 225	45.313	62.285

Flooding and MOCA have the highest packet overhead per successful request. CACMAN with lazy replies of 640ms has the least in both variations as shown in Table VIII.

5.5. Concluding Remarks

From the previous discussion, we summarize our simulation results into the following points:

1) The effect of the threshold is noticeable, and the more it increases, the less the likelihood of getting all partials from the first attempt.

2) The number of replies received per request is still larger than required except when $t = 25$, in which case the number of successful replies is reduced dramatically. However, when we applied the X-replies variation, the number of successful CREPs increased noticeably.

3) The mobility effect is not that significant since CACMAN utilizes local broadcasts, and mobility will have a significant effect when routing information is involved in the protocol.

4) Cache size obviously has an impact on performance. However, going from a cache size of 75 to 150 is more significant and effective than going from 150 to 225, and this is consistent in other thresholds and max speeds, which suggests 150 as the best compromise between cache size and hit ratio.

5) Techniques like Cache-Focus and Lazy Replies with Cancellation can increase the performance of CACMAN without noticeable increase in overhead.

6) CACMAN has much faster response time (about 16 times faster) as compared to flooding and MOCA due to the fact that some partials are available at the local cache of the requester. Besides, CACMAN requests never propagate more than one hop.

SECTION 6
CONCLUSION AND FUTURE WORK

This section is the conclusion of this dissertation. We will summarize the motivation for our work, the problem we addressed, and our proposed solution to offering secure and highly available CA service in MANET. We will then present a roadmap for enhancing and extending our work in the future. The section is organized into two subsections: Section 6.1 is the summary and conclusion of this work, and Section 6.2 is an elaboration about further extensions and enhancements.

6.1. Summary and Conclusion

The introduction of MANET has reshaped our thinking about many problems that researchers have faced in computer systems that caused them to seek genuine solutions due to MANET's unique and distinctive characteristics such as dynamic topology changes combined with power restrictions, transmission range, and limited computational capabilities.

In this dissertation, we started by showing that security is a fundamental part of MANET, yet it has not been seriously addressed until recently. Every aspect of MANET is inherently vulnerable to wider security attacks compared to wired networks. Frequency jamming, routing protocol disruption, snooping, or straight physical compromising of nodes (e.g. [41], [57], [73], [28]) are just a few examples of some security problems that have a wide spectrum of possible solutions in MANET. Quality of Service (QoS), too, has been tackled in MANET from a variety of perspectives (e.g. [13], [14], [27], [45]), and guaranteeing availability of vital services is yet another difficult MANET problem that requires specialized solutions for this environment.

Public-Key Infrastructure (PKI) is an important service that provides a framework for a system using public-key primitives and operations to achieve security functions such as Authentication, Authorization, Privacy, Integrity, and Non-repudiation.

Certificates, as a major PKI component, are used for encrypting and/or signing many vital applications, such as authentication, exchange of routing info, encrypting or signing emails, and much more. A Certificate Authority (CA), at the heart of any PKI system, is usually used to organize, store, and issue those certificates. Furthermore, adopting PKI in MANET is not an easy task since providing a reliable and secure service is much more challenging as compared to well-connected and reliable wired networks. We discussed solutions proposed to adapt CA in MANET (Section 3), and most of these solutions rely on secret sharing mechanisms (Section 2.4) to increase security and availability. These solutions assume installing the CA service in just one node will make it vulnerable, and an exact replication of the CA will make the situation even worse.

Kong *et al.* proposed Ubiquitous and Robust Security Architecture (URSA) (Section 3.3) to provide pervasive CA services by making all (n) nodes in the network share CA functionality. The private-key of the service is shared by all nodes, but any t member can form a coalition to sign certificates which are trusted by the whole network. The problem of choosing a low threshold value to guarantee availability, in addition to other technical issues, reduces its practicality.

We described Mobile CA (MOCA) (Section 3.5) protocol and showed that it has been introduced as the counterpart of the Cornell Online Certificate Authority (COCA)(Section 3.4) framework, but with consideration to MANET's unique properties. MOCA is a mobile node in the MANET, and it is selected to provide distributed CA functionality. When a client wants to obtain a certificate, it sends Certificate Request (CREQ) packets to at least t (usually more) MOCA servers and waits for a reply. Any MOCA that receives a CREQ will reply by sending a Certification Reply (CREP) packet containing its partial signature. Once the client receives at least t valid CREPs, it can reconstruct the whole certificate. CREQ and CREP have been embedded in Route Request (RREQ) and Route Reply (RREP) messages that are found in on-demand ad-hoc routing protocols like AODV [63] and DSR [16] to reduce the amount of overhead packets. MOCA increases availability by letting clients send more than t unicast CREQ just in case. Additionally, to avoid flooding the network, MOCA clients inspect the route

table to see if any cached routes to MOCAs are available, making flooding a last resort when the cached routes are less than a given threshold.

We have two major concerns about MOCA which we believe may reduce the usability of the framework. The first concern is that the number of MOCA nodes is relatively high, about 10 to 20% of the total number of participants. We believe that this does not come without some consequences, such as CA synchronization and key refreshing overhead. The second concern is that the number of control packets, the sum of CREQ and CREP, generated by MOCA is high compared to Flooding. In the best case, MOCA saves about 30% of packet overhead when $t = 5$ and only about 5% when $t = 25$.

We started our work, "CAching Certificates in Mobile Ad-hoc networks" (CACMAN), by a preliminary evaluation using Monte Carlo simulation presented in [8] but without any consideration of mobility. Then, we revised our work [9], [10] and used a more realistic simulator i.e. NS-2 [47]. The aim of our research in this dissertation is to minimize the burden of adapting CA services in MANET by minimizing the packet overhead and maintaining high availability of the service at the same time. This has been achieved by making clients share some responsibilities with CA servers by *cooperatively* caching some certificates generated by CA servers. We showed how the characteristics of certificates have made caching a reasonable solution to the availability problem. Our cashing-based framework has addressed the security and performance challenges of providing CA services in MANET. Moreover, we suggested techniques with minimal overhead that help achieve our main goal, availability, without compromising MANET security. We showed the feasibility of our framework and compared it to other related systems using the NS-2 simulator which gave us more insight into the problem from the performance perspective.

In CACMAN, each client that wants to obtain the certificate of other client(s) to establish a secure communication or to encrypt some messages will search its own cache for a complete certificate of the subject. If found, then the request is completed; otherwise, it will identify the missing partials and proceed by making a local broadcast, but if it happens that it has a short route to the subject in question, it sends the CREQ to

that subject instead of making a local broadcast. Every client that receives a CREQ message inspects its cache for a possible hit. It sends a CREP back to the sender if a full or partial certificate(s) is found for the subject. After that, the client waits for a specified time; if it receives sufficient responses then it reconstructs the certificate. If the request initiator times out because insufficient partials have been received or the reconstruction has failed, then the client may try to contact CA servers by flooding the network or by sending unicast messages if it has sufficient routes to them. Otherwise, it reports a failure to the client or retries later.

In our simulations, we used the most recent build of NS-2 to simulate the same scenario used in MOCA to ensure fair comparisons. Each configuration was replicated at least three times. For CACMAN protocol, we used four cache sizes – (75, 150, 225, and 300 share slots) - and four thresholds for key construction – (5, 15, 20, and 25). The results show CACMAN can enhance the CA service, and we summarize our findings into the following points:

1) The threshold had noticeable effect on performance. Higher thresholds, though increasing system security, will make the likelihood of getting all partials from the first attempt lower. And to a great extent, CACMAN is able to tolerate high threshold better than MOCA.

2) The number of replies received per request is still larger than required except when $t = 25$, in which case the number of successful replies is reduced dramatically. However, when we applied the X-replies variation, the number of successful CREPs increased noticeably.

3) Mobility does not have a significant effect on CACMAN performance due to its utilization of local broadcasts to obtain service.

4) Cache size plays a major role in CACMAN. Increasing the size from 75 shares to 150 is significant and effective. However, going from 150 to 225 has lesser impact, and this holds regardless of the threshold and/or maximum speeds, which makes 150 ideal in terms of cache size and hit ratio tradeoffs for the examined environment.

5) CACMAN has a very significant reduction in packet overhead; for example, MOCA saves about 30% over Flooding when threshold is set to 5, while CACMAN saves 92% at the same threshold. Furthermore, the overhead is also affected proportionally, in the case of CACMAN, by the cache size and the number of replies.

6) Various proposed enhancements have noticeable impact on CACMAN performance. X-Replies, Lazy-Replies with Cancellation, and Cache-Focus played a pivotal role in enhancing the framework availability.

7) CACMAN has faster response time (up to 16 times) than flooding and MOCA due to its caching techniques; moreover, CACMAN requests never propagate more than one hop.

8) The framework's ability to maintain high availability with much less network overhead allows many other protocols that depend on public-key operations to be introduced in MANET without the fear of increasing the overall overhead caused by security.

We have investigated other topics associated with our framework such as the impact of removing CA servers from CACMAN without significant loss in performance. The impact of CA key refreshing on the framework and how to overcome the consequences thereof have also been investigated. "Partial certificates compaction and redundancy removal" and "cache replacement policies" have been conceptually discussed to complement our framework. We also analyzed the impact of applying different revocation schemes. Enhancements to the basic protocol such as Cache-Focus and Lazy-Replies with Cancellation has proven to be useful to boost the framework further. Finally, CACMAN deployment in hybrid environments has been discussed and shown to be useful as well.

In conclusion, this dissertation has the following major contribution:

CACMAN is able to reduce network overhead associated with threshold based CA service up to 92% as compared to related work with very short

response time (about 16 times faster). The dependency on CA servers has been reduced and CACMAN is able (in the simulation) to tolerate as much as 20 inoperative CA servers out of 30 (due to crash or offline operations) without noticeable decrease in the performance. By separating availability from security, which is an inherent problem of threshold-based solutions and in CACAMAN, we are able to alleviate that problem.

6.2. Future Extensions

Here, we will present different possible directions that could be investigated for further expansion of our work in this dissertation.

6.2.1 New Communication Paradigm and Applications

In this thesis, our focus and discussion was in the domain of caching certificates and how to speedup this operation with less overhead. If we take a closer look into the CACMAN request operation, the sequence of a transaction will always follow a one-to-many-to-one [22] communication paradigm. We call the first part of the previous term (i.e. one-to-many) "Quorum-cast" since we are trying to reach a threshold of servers. Quorum-cast is considered a generalization of multicasting, and more details can be found in [29], [24]. Carter *et al.* [22] have discussed some other applications, besides MOCA (Section 3.5), for one-to-many-to-one, such as version 4 of the Network Time Protocol (NTP) [50].

The introduction of a caching component, as we did in CACMAN, could bring more efficiency into one-to-many-to-one types of applications since clients collaboratively help the servers. However, not all application are cache compatible, for example, consider NTP where it is makes no sense to cache servers' replies. Finding other potential applications that could benefit from our introduction of caching as part of the paradigm is one of our future goals.

6.2.2 Low-density Topologies

Section 5 is dedicated to examine the performance part of CACMAN and compare it to related work. However, we did not have much freedom to control many of the simulation

parameters because a fair comparison is impossible unless we test both protocols under identical conditions.

Thus, it is interesting to investigate on performance of CACMAN into different topologies and node sizes. The topology we used in studying CACMAN is a $1000m^2$ area. Although this size and the total number of nodes (i.e. 150) is common in MANET simulations, we think it is relatively dense. We think that it is worth extending the simulation to cover larger, hence lower-density areas with the same number of nodes scenarios so that we can see how to tune CACMAN to keep its high performance. Our preliminary investigations show that the performance of CACMAN is expected to improve if we allow a CREQ to propagate a little bit more than what is currently allowed (e.g. TTL = 2 instead of local broadcast). Another technique worth trying is to use some probabilistic forwarding scheme (i.e. the CREQ is forwarded to next hop with a given probability) to find a balance point between maximum hit ratio and minimum overhead. However, possibly other techniques like those discussed in Section 4.4 could lead to different results.

6.2.3 Certificate Usage Distribution and Scalability

In this dissertation, we assumed that all certificates have similar popularity and are distributed uniformly in client caches. This was also the case when a CACMAN client makes a CREQ in which the probability of requesting a certificate is uniformly distributed. However, this could not be the case in practice because of reference locality (e.g. temporal or geographical). Although some empirical studies regarding the demand of multimedia objects show that accessing patterns follow Zipf distribution [45], we cannot say that is the case for certificates, and to the best of our knowledge, such a study is not available. Investigating certificate access and usage patterns in large scale enterprises or on the Internet scale is an important topic to be studied. We think that knowing how certificates are used and exchanged between principals will enable us to make better plans for cache sizes and most likely will reduce the amount of memory required as compared to the uniform distribution that we have assumed in this dissertation.

Another benefit of finding a distribution model for certificate requests is scalability since the cache size usually cannot cope with the dramatic increase of the number of existing certificates and nodes. Currently, CACMAN memory requirements increase linearly with the number of existing certificates, given that we have fixed all other parameters. It is known that the lack of locality will adversely affect the scalability of any application that utilizes caches to increase its performance.

6.2.4 Analytical Model

Another interesting issue is building an analytical model that captures the most important parameters involved in the framework. Such a model will enable us to find optimum values among conflicting parameters to satisfy the application requirements. In CACMAN, we think building a generic topology-independent analytical model is worth being investigated. For example, a model that takes parameters such as cache size, total number of objects (i.e. certificates), threshold, objects distribution, and the number of accessible neighbors into consideration to find, say, the hit ratio.

6.2.5 Miscellaneous Issues

Here, we briefly list other noteworthy issues. Some are considered easy from the implementation point of view, but the problem of finding the right assumptions may be not that simple.

1) *Cross-Layered*[20] *Secure Framework*: It is interesting to investigate how to integrate CACMAN in a framework that addresses all identified security threats in MANET and provide a single practical solution for them. Such framework is security-centric, where security features and sub-protocols exist in every layer of MANET stack. We urge researchers to study the possibility and problems which may arise by such integration.

---

[20] M. Conti et al. brought the issue of the importance of cross-layering in MANET, and in their model, security indeed is a vertical element of the MANET stack [23].

2) *Integration of Detection System*: As a component of the previous point, misbehaving nodes can decrease the usability of CACMAN by injecting bad partials. Using verification techniques, honest CACMAN clients can discover such partials, and they may report the abuse incidents to CA servers, and if the CA collects many complains about the same client, it may revoke the certificate of the malicious node and broadcast such a decision. It is interesting to investigate the details of this protocol.

3) *Generalization of the Underlying Network Infrastructure*: CACMAN has been designed with MANET in mind. However, it is possible to extend it to other environments such as Internet level peer-to-peer overlay networks which is a good candid environment, though the parameters will be adapted accordingly.

4) *CRL Performance*: It is important to evaluate the performance impact of various CRL schemes when applied into MANET environments to see how long it takes to disseminate CRL information to the entire network.

5) *Practical Key Refreshing Time for CA Servers*: Key refreshing has inherent tradeoff between communication overhead, i.e. at least $O(n^2)$, and the vulnerability created by allowing the intruder more time to compromise the system if the window of vulnerability is too long. Knowing the impact of choosing different key refreshing intervals on CACMAN performance is important since it will affect the freshness of cached results, as we discussed it in Section 4.7.1.

6) *The Efficiency of Exchanging Caches Contents Procedure*: This point could be tackled based on the findings obtained from the previous point. Key-refreshing frequently will allow many incompatible shares issued in different epochs to exist in CACMAN clients' caches, which means additional overhead. Using the protocol described in Section 4.7.2 would alleviate the situation, and it will be interesting to get a quantitative result for the procedure performance.

REFERENCES

[1] Bluetooth consortium [online]. Available:http://www.bluetooth.org/

[2] *IEEE 802.11 Handbook: A Designer's Companion*, 2nd Edition, Feb. 2005.

[3] Intel XScale Microarchitecture [online], Technical Summary; http://www.intel.com.

[4] The Online Certificate Status Protocol (OCSP); http://www.baltimore.com /devzone/pki/ocsp.html

[5] C. Adams and S. Farell, "IETF RFC 2510 Internet X.509 Public Key Infrastructure Certificate Management Protocols," Mar. 1999; http://www.ietf.org/rfc/rfc2510.txt

[6] C. Adams, S. Llyod, and S. Kent, *Understanding PKI: Concepts, Standards, and Deployment Considerations*, pp. 28, 2nd Edition, Addison-Wesley Professional, Nov. 2002.

[7] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–116, Aug. 2002.

[8] L. Al-Sulaiman and H. Abdel-Wahab, "Cooperative Cashing techniques for increasing the availability of MANET Certificate Authority services," *the 3rd ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-05)*, Cairo, Egypt, Jan. 2005.

[9] _____, "CACMAN: a Framework for Efficient and Highly Available CA Services in MANETs," *the 10th IEEE Symposium on Computers and Communications (ISCC 2005)*, Cartagena, Spain, June 27-30. 2005.

[10] _____, "Adapting CA services in MANET," Technical Report #oai_dc_odu_TR_04_01, Computer Science Department, Old Dominion University, Dec. 2004.

[11] N. Asokan and P. Ginzboorg, "Key Agreement in ad hoc networks, Computer Communications," vol 23, pp. 1627-1637, 2000.

[12] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward, "A Distance Routing Effect Algorithm for Mobility (DREAM)," *ACM/IEEE MobiCom'98*, Dallas, Texas, USA, Oct. 1998.

[13] A. Bestavros and S. Jin, "OSMOSIS: Scalable Delivery of Real-Time Streaming Media in Ad-Hoc Overlay Networks," *In Proceedings of IEEE ICDCS'03 Workshop on Data Distribution in Real-Time Systems*, Providence, RI, USA, May 2003.

[14] L. Bononi, L. Budriesi, D. Blasi, V. Cacace, L. Casone, and S. Rotolo, "A differentiated distributed coordination function MAC protocol for cluster-based wireless ad hoc networks," *Proceedings of the 1st ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pp. 77 – 86, Venezia, Italy, Oct. 2004.

[15] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 85–97, Dallas, TX, Oct. 1998.

[16] J. Broch and D.B. Johnson, "The Dynamic Source Routing Protocol for Mobile Ad-hoc Networks," *IETF Internet Draft*, Feb. 2003.

[17] S. Buchegger and J. –Y. Le Boudec. "Performance analysis of the CONFIDANT protocol," *In Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 226–236, June 2002.

[18] L. Buttyán and J. –P. Hubaux (Eds.), "Report on a Working Session on Security in Wireless Ad Hoc Networks", *SIGMOBILE Mobile Computing and Communications Review*, vol. 7, no. 1, pp. 74-94, New York: ACM Press, 2003.

[19] L. Buttyán and J. –P. Hubaux, "Stimulating cooperation in self-organizing mobile ad hoc networks," *ACM/Kluwer Mobile Networks and Applications (MONET)*, 2002.

[20] S. Čapkun, J. Hubaux, and L. Buttyán, "Mobility Helps Security in Ad Hoc Networks," *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, 2003.

[21] C. Carter, S. Yi, and R. Kravets, "ARP Considered Harmful: Manycast Transactions in Ad Hoc Networks," *IEEE WCNC*, 2003.

[22] C. Carter, S. Yi, P. Ratanchandani, and R. Kravets, "Manycast: exploring the space between anycast and multicast in ad hoc networks," *In Proceedings of the 9th annual international conference on Mobile computing and networking*, pp. 273–285, ACM Press, 2003.

[23] M. Conti, G. Maselli, and G. Turi, "Cross-Layering in Mobile Ad Hoc Network Design," *Computer Magazine*, pp. 48-51, Feb, 2004.

[24] S. Y. Cheung and A. Kumar, "Efficient Quorumcast Routing Algorithms," *INFOCOM*, pp. 840-847, 1994.

[25] Y. Desmedt, "Society and group oriented cryptography: A new concept," *Advances in Cryptology Crypto '87 proceedings*, C. Pomerance, ed., no. 293, pp. 120-127, in LNCS, Santa Barbara, California, U.S.A., Springer-Verlag, 1988,.

[26] Y. Desmedt, "Some Recent Research Aspects of Threshold Cryptography," *Information Security, First International Workshop ISW '97*, 1997.

[27] M. C. Domingo and D. Remondo, "An interaction model between ad-hoc networks and fixed IP networks for QoS support," *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pp. 188 – 194, Venice, Italy, Oct. 2004.

[28] J. R. Douceur, "The Sybil Attack," IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems, pp. 251-260, London:Springer-Verlag, 2002.

[29] B. Du, J. Gu, D.H.K. Tsang and W. Wang, "Quorumcast Routing by Multispace Search," *Proc. of IEEE Globecom'96*, pp. 1069-1073, London, UK, Nov. 1996.

[30] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen, "SPKI certificate theory," *Internet RFC 2693*, Sept. 1999.

[31] R. Friedman, "Caching Web Services in Mobile Ad hoc Networks: Opportunities and Challenges," in *Proceedings of the 2002 Workshop on Principals of Mobile Computing (POMC 2002)*, Toulouse, France, Oct. 2002.

[32] Z. J. Haas, J. Deng, B. Liang, P. Papadimitratos, and S. Sajama, "Wireless Ad-hoc Networks," *Encyclopedia of Telecommunications*, J. Proakis, editor, John Wiley, 2002.

[33] Y. -C. Hu, D. B. Johnson, and A. Perrig, "Secure efficient distance vector routing in mobile wireless ad hoc networks," in *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, June 2002.

[34] Y. -C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," in *Proceedings of the 8th ACM International Conference on Mobile Computing and Networking (MobiCom)*, Sept. 2002.

[35] Y. -C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: A defense against wormhole attacks in wireless ad hoc networks," Technical Report TR01-384, Department of Computer Science, Rice University, Dec. 2001.

[36] J. -P. Hubaux, L. Buttyán, and S. Čapkun, "The quest for security in mobile ad hoc networks," *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Long Beach, CA, USA, Oct. 2001.

[37] P. Jacquet, P. Muhlethaler, and A. Qayyam, "Optimized Link State Routing Protocol," *IETF MANET*, Internet Draft, Nov. 1998.

[38] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, John Wiley and Sons, April 1991.

[39] S. Jarecki, "Proactive secret sharing and public key cryptosystems," master thesis, MIT, 1995.

[40] J. G. Jetcheva and D. B. Hohnson, "Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks", *In proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC '01)*, Long Beach, CA, Oct. 2001.

[41] C. Kaufman, R. Perlman, and M. Speciner, *Network security: Private Communication in a public world*, Prentice Hall, 2nd edition, 2002.

[42] Y. B. Ko and N. H. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks," *ACM/IEEE MobiCom'98*, Dallas, Texas, USA, Aug. 1998.

[43] 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 ICNP '01*, pp. 251-260, 2001.

[44] P. Kyasanur and N. H. Vaidya, "Detection and handling of MAC layer misbehavior in wireless networks," technical report, UIUC, Aug. 2002.

[45] W. Lau, M. Kumar, and S. Venkatesh, "Mobile Ad Hoc Networks: A cooperative cache architecture in support of caching multimedia objects in MANET," in *Proceedings of the 5th ACM international workshop on Wireless mobile multimedia*, Sept. 2002.

[46] D. Malki and M. Reiter, "Byzantine Quorum Systems," *Proceedings of the 29th ACM Symposium on Theory of Computing*, May 1997.

[47] S. McCanne and S. Floyd, The LBNL network simulator (NS-2) [online]. Available: http://www.isi.edu/nsnam/ns/.

[48] P. McDaniel and A. Rubin, "A response to Can we eliminate revocation lists?", in *Proceedings of Financial Cryptography, the Fourth International Conference (FC'00)*, Lecture Notes in Computer Science, vol. 1962, pp. 245-258, Y. Frankel, (Ed.), Springer-Verlag, Berlin, 2000.

[49] S. Micali, "NOVOMODO: Scalable Certificate Validation and Simplified PKI Management," in *Proceedings of the 1st Annual PKI Research Workshop*, NIST, Gaithersburg MD, USA, April 2002.

[50] D. Mills, "The network time protocol (NTP) distribution [online]". Available: http://www.eecis.udel.edu/~mills/ntp/html/manyopt.html.

[51] MITRE Corporation, *Final Report: Public Key Infrastructure*, tech. report, National Institute of Technology, 1994.

[52] R. Mukkamala, S. Das, and M. Halappanavar, "Recertification: A Technique to Improve Services in Public-key Infrastructure," *16th Annual IFIP WG 11.3 Conference on Data and Application Security*, University of Cambridge, UK, July 2002.

[53] S. Murthy, J. J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks," *ACM Mobile Networks and Applications Journal*, Oct. 1996.

[54] M. Myers, "Revocation: Options and challenges," in *Proceedings of Second International Conference on Financial Cryptography (FC'98)*, Vol. 1465, pp. 165-171, Lecture Notes in Computer Science, Springer-Verlag , Feb. 1998.

[55] S. Y. Ni, Y. C. Tseng, Y.S. Chen, and J. P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *ACM/IEEE MobiCom*, 1999.

[56] S. Olariu, A. Wadaa, L. Wilson, and M. Eltoweissy, "Wireless Sensor Networks Leveraging the Virtual Infrastructure," IEEE Network, pp. 51-56, July 2004.

[57] R. Ostrovsky and M. Yung, "How to withstand mobile virus attacks," in *Proceedings of the 10th Annual Symposium on Principals of Distributed Computing (PODC'91)*, pp. 51–59, Montreal, Quebec, Canada, Aug. 1991.

[58] E. Pagani and G. P. Rossi, "A framework for the admission control of QoS multicast traffic in mobile ad hoc networks," in *Proceedings of the 4th ACM International Workshop on Wireless Mobile Multimedia*, pp. 2-11, July 2001.

[59] P. Papadimitratos and Z. J. Haas, "Secure Routing for Mobile Ad hoc Networks," *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, San Antonio, TX, Jan. 2002.

[60] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Sensor Networks," *Communications of the ACM*, May 2000.

[61] T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," *Advances in Cryptology (CRYPTO'91)*, LNCS, vol. 576, pp. 129-140, Springer-Verlag, 1992.

[62] M. R. Pearlman and Z. J. Haas, "Determining the Optimal Configuration for the Zone Routing Protocol," *IEEE Journal on Selected Areas in Communications*, vol.17, no. 8, pp. 1395-1414, Aug. 1999.

[63] E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing," in *The Second IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90-100, New Orleans, LA, Feb. 1999.

[64] R. Rivest, "Can We Eliminate Certificate Revocation Lists?," in *Financial Cryptography, the Second International Conference (FC'98)*, 1998.

[65] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer, "A secure routing protocol for ad hoc networks," in *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP)*, Nov. 2002.

[66] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, Nov. 1979.

[67] A. Tanenbaum, *Computer Networks*, 3rd edition, Prentice Hall, 1996.

[68] D. Wetherall, "OTcl Tutorial Version 0.96[online]," Sept. 1995. Available: ftp://ftp.tns.lcs.mit.edu/pub/otcl/doc/tutorial.html

[69] J. Yoon, M. Liu, and B. Noble, "Random Waypoint Considered Harmful," *IEEE Infocom 2003*, San Franciso, CA, 2003.

[70] S. Yi and R. Kravets, "MOCA: Mobile Certificate Authority for Wireless Ad-hoc Networks," *2nd Annual PKI Research Workshop (PKI03)*, Apr. 2003.

[71] S. Yi and R. Kravets, "Practical PKI for Ad Hoc Wireless Networks," University of Illinois at Urbana-Champaign, Tech. Rep. UIUCDCS-R-2002-2273/UILU-ENG-2002-1717, May 2002.

[72] M. Zapata and N. Asokan, "Securing ad hoc routing protocols," in *Proceedings of the ACM workshop on Wireless Security (WiSE '02)*, pp. 1-10, New York: ACM Press, 2002.

[73] L. Zhou and Z. J. Haas, "Securing Ad-hoc Networks," *IEEE Network Magazine*, Nov. 1999.

[74] L. Zhou, F. Schneider, and R. van Renesse, "COCA: A secure distributed on-line certification authority," *ACM Transactions on Computer Systems*, vol. 20, no. 4, pp. 329-368, Nov. 2002.

[75] Y. Zhang and W. Lee, "Intrusion detection in wireless ad-hoc networks," in *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom)*, Aug. 2000.

[76] Y. Zhang, W. Lee, and Y. Huang, "Intrusion detection techniques for mobile wireless networks," *ACM/Kluwer Mobile Networks and Applications (MONET)*, 2002.

APPENDIX A
NS-2 SIMULATOR

A.I Network Simulator (NS-2)

"NS is a discrete event simulator targeted at networking research. NS provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. NS began as a variant of the REAL[21] network simulator in 1989 and has evolved substantially over the past few years. In 1995 NS development was supported by DARPA through the VINT[22] project at LBL, Xerox PARC, UCB, and USC/ISI. Currently NS development is support through DARPA with SAMAN and through NSF with CONSER, both in collaboration with other researchers including ACIRI. NS has always included substantial contributions from other researchers, including wireless code from the UCB Daedelus and CMU Monarch projects and Sun Microsystems" [47].

NS-2 has been implemented using OTCL [68] and C++ programming languages. OTCL is an object-oriented variant of TCL (Tool Command Language) and it is designed for writing scripts and configurations files rapidly and without any necessary compilation but with the shortcoming of execution speed since it is an interpreted language. C++ is used for writing NS-2 core, protocols, and any code that does not change frequently. Both languages are linked using some API that allows both languages to see each other variables and objects at run time.

---

[21] See http://www.cs.cornell.edu/skeshav/real/overview.html.

[22] See http://www.isi.edu/nsnam/vint/index.html.

A.II Technology and Environment Used in CACMAN Implementation

We were able to build NS-2 version 2.27 successfully (after some effort) at the ODU Computer Science Department on two Sun Fire™ V880 (each equipped with 8 UltraSPARC processors, 32GB RAM, and Solaris™ 9 OS). We have followed an Object Oriented Design approach to develop the framework using C++ and benefited from its Standard Template Library (STL) though it was not recommend to be used in NS for portability reasons. GCC 3.3.2 was used to compile about 2000 lines of source code for the simulation and configuration. Perl 5.8.0 was used to analyze and aggregate 67 GB of trace file generated by our simulation. MS-Office™ 2003 suite used to plot figures, draw diagrams, and typesetting the dissertation.

APPENDIX B
TECHNICAL OVERVIEW OF CACMAN

This appendix contains snapshots from CACMAN design and implementation. Static diagram is shown in Appendix B.I. Appendix B.II contains packet types and formats for some CACMAN messages. Appendix B.III is the OTCL scripts for configuring and running the simulation. Finally, Appendix B.IV contains C++ code of a one of the CACMETN core methods.

B.I Class Diagram

Fig. 22 is Unified Modeling Language (UML) static[23] representation of CACMAN design and it summaries some important facts about the work. We have implemented other protocols as a proof of concept prototypes. Class "Agent" is the root that all other protocol will derive from and it contains some virtual methods that may be overridden by each protocol. In our simulation the CA servers are just MOCA servers and they do not playing in additional rule besides sending CREPs whenever they receiver any requests. CACMAN client has an important member, i.e. "sharesCache", of type "CertCache" which is responsible for managing partials.

[23] The Dynamic Model of the system is partially shown in Section 4.3

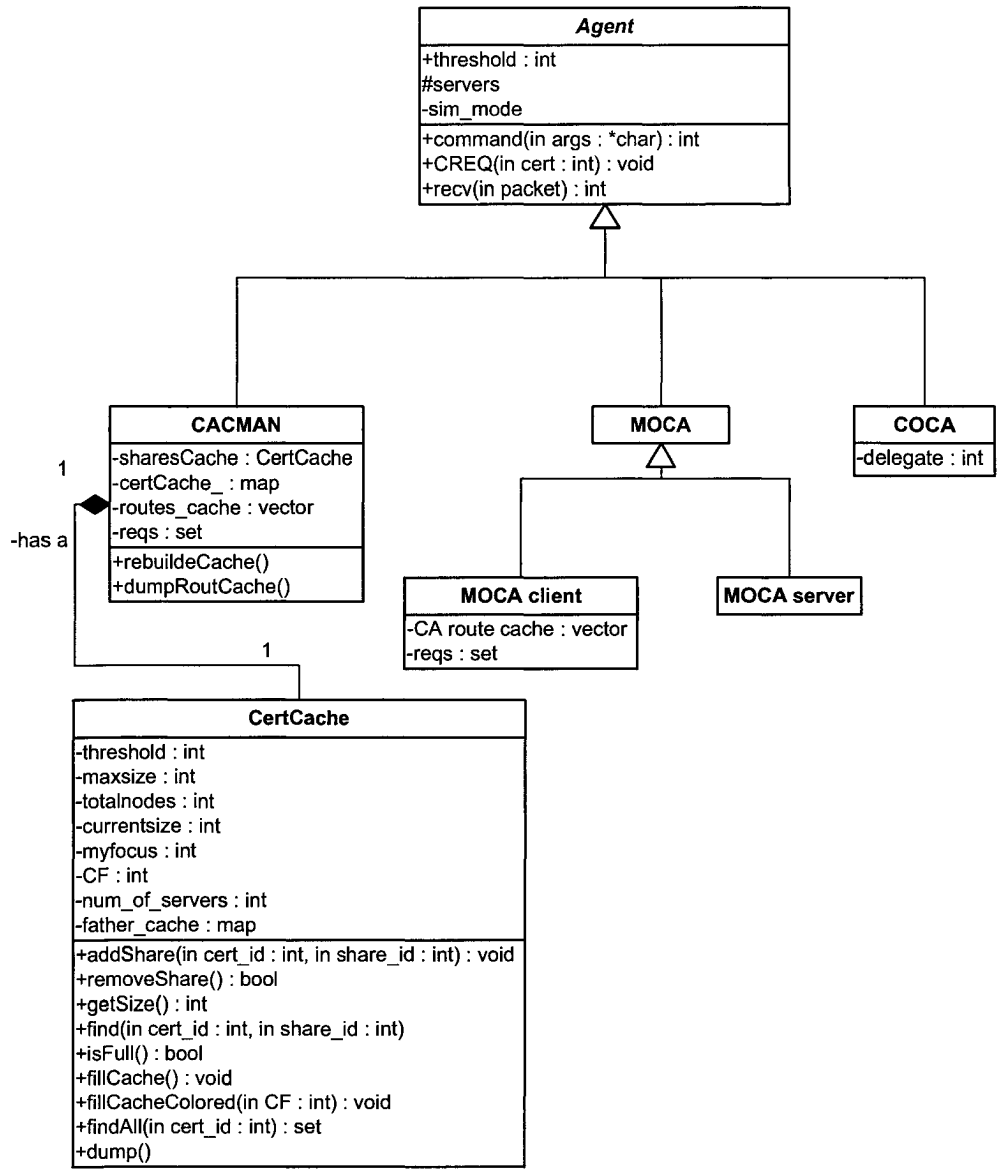


Fig. 22. Static structure of important CACMAN modules

We also summarize important methods and properties in Table IX for a better understanding of Fig. 22.

TABLE IX
METHODS AND PROPERTIES SUMMERY

Method or property	*Purpose*
Agent::threshold	Security threshold of the system.
Agent::servers	Holds the addresses of CA servers.
Agent::sim_mode	Node type (e.g. CACMAN, MOCA, ...etc).
Agent::Command	Receives commands passed from OTCL in string format. The method should be overridden and each protocol has its own set of commands.
Agent::CREQ	Certificate Request expects principal ID as a parameter. The method should be handled by each protocol according to its technique of sending CREQs
Agent::recv	Packets received by the agent, usually CREQ, are passed to this method. The packet contains the protocol header besides IP header.
CACMAN::sharesCache	This object is the cache manager of CACMAN agent and more details about its contract is shown under CertCache below.
CACMAN::certCache_	A hash table for outstanding requests
CertCache::threshold	The cache manager needs to now the threshold since this information is important for share combining.
CertCache::maxsize	Maximum size for the cache measured by number of shares.
CertCache::currentsize	Contains the current number of cached shares.
CertCache::myfocus	A set that contains ID numbers of CAs which this client will focus on (used in Cache-Focus enhancement only)
CertCache::CF	The size of the cache focus group.
CertCache::father_cache	It is the superset that contains all possible shares. Its replications are used for initializing new caches.
CertCache::num_of_servers	Used by "father_cache"
CertCache::addShare	Adds a share to CACMAN client cache.
CertCache::removeShare	Remove a share from CACMAN client cache. The manger will the algorithm specified in Section 4.7.4 for choosing the victim.
CertCache::find	Search for a specific certificate.
CertCache::isFull	Returns true if the certificate cache is full.
CertCache::fillCache	Initializes the cache randomly with partials based on some constraints (e.g. number of servers, size, and CF usage)
CertCache::fillCacheColored	Called by "fillCache" when cache-focus is in use.
CertCache::findAll	Returns a set that contains the shares of the specified certificates.
CertCache::dump	Print cache contents for debug purpose.

B.II CACMAN Packet Format

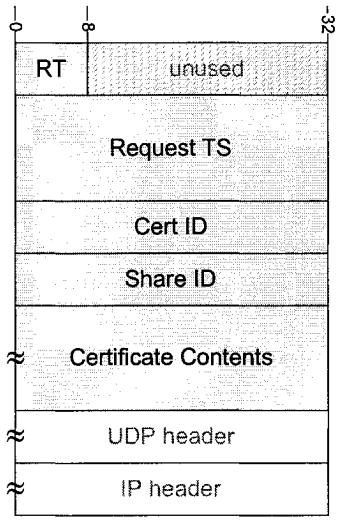


Fig. 23. CACMAN's packet format. RT stands for request type (CREQ=0, CREP=1, RCancel=2). Request TS is to hold the timestamp when the CREQ is issued and used for setting various timers and statistics. Cert ID is used to identify the CREQ and to tell what certificate is inside the payload if used in CREP messages. Share ID is to tell which share is in the CREP payload. Certificate Contents holds certificate data. The implementation uses a slight modified design of this format.

B.III An Example of a Simulation Scenario

Fig. 24 shows CACMAN main script used for running NS-2 simulation.

```
# cacman.tcl
# ======================================================================
# Laith Al-Sulaiman (lal-sula@cs.odu.edu)
# Old Dominion University, Computer Science Department
# 2005
# ======================================================================

# Define options and simulaiton environments

set val(chan)       Channel/WirelessChannel
set val(prop)       Propagation/TwoRayGround
set val(netif)      Phy/WirelessPhy
set val(mac)        Mac/802_11
set val(ifq)        Queue/DropTail/PriQueue
set val(ll)         LL
set val(ant)        Antenna/OmniAntenna
```

Fig. 24. CACMAN main script written in OTCL.

```
set val(x)              1000    ;# X dimension of the topography
set val(y)              1000    ;# Y dimension of the topography
set val(ifqlen)         50      ;# max packet in ifq
set val(seed)           0.0     ;
set val(adhocRouting)   AODV    ;
set val(nn)             20      ;# default number of nodes
set val(thresh)         5       ;# default threshold
set val(cp)             "scens/moca-sen.tcl" ;
set val(sc)             "scens/scen-1000x1000-20-0-10-1" ;
set val(stop)           600     ;# simulation time
set val(logAgent)       ON      ;

# ======================================================================
# Main Program
#
# ======================================================================

# new seed and all streams will be derived from that seed
global defaultRNG
$defaultRNG seed 10877


# parse the command line
for {set i 0} {$i < $argc } { } {
   set [lindex $argv $i]  [lindex $argv [expr $i+1]]
   set i [expr $i + 2]
}

# read the total number of nodes, threshold, and scenario file
if {[info exists n]} {
  set val(nn) $n
}

if {[info exists t]} {
  set val(thresh) $t
}

if {[info exists sc]} {
  set val(sc) $sc
}

# set number of servers
if { $val(nn) >= 150 } {
  set val(servers) 30
} else {
  set val(servers) $val(thresh)
}

# in case of we have 300 nodes make val(servers) = 50
if { $val(nn) >= 300 } {
  set val(servers) 50
}
```

Fig. 24. Continued.

```
puts "servers = $val(servers)\nthreshold = $val(thresh)\nnnn =
$val(nn)\nfile = $val(sc)"


# Initialize Global Variables
#
# define simulation mode (CACMAN, MOCA, COCA, KONG) and threshold

set simulation_mode "CACMAN"
set threshold $val(thresh)

# create simulator instance

set ns_            [new Simulator]

# setup topography object

set topo    [new Topography]

# prepare the trace file which is derived from the command line params
set tracefd [open "$val(sc)-($simulation_mode-$val(thresh)).tr" w]
$ns_ use-newtrace
$ns_ trace-all $tracefd

# define topology
$topo load_flatgrid $val(x) $val(y)

# Create God
set god_ [create-god $val(nn)]

#
# define how node should be created
#

# this proc is to attach nodes to the trace descriptor
proc addTracer { agt } {
        global ns_
      global tracefd
      global val

      if { $val(logAgent) == "ON" } {
            set T [new Trace/Generic]
            $T target [$ns_ set nullAgent_]
            $T attach $tracefd
            $T set src_ 1
            $agt log-target $T
      }
      return 1
}

#global node setting
```

Fig. 24. Continued.

```
$ns_ node-config -adhocRouting $val(adhocRouting) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -channelType $val(chan) \
                   -topoInstance $topo \
                     -agentTrace $val(logAgent)  \
                 -routerTrace ON\
                 -macTrace OFF \
                     -movementTrace OFF


#
#  Create the specified number of nodes [$val(nn)] and "attach" them
#  to the channel.

for {set i 0} {$i < $val(nn) } {incr i} {
      set node_($i) [$ns_ node]
      $node_($i) random-motion 0          ;# disable random motion
}


#
# Define traffic model
#
puts "Loading connection pattern from $val(cp) ..."
source $val(cp)

# Define node movement model
#
puts "Loading scenario file from $val(sc) ..."
# the content of that source is shown in the next figure
source $val(sc)

#
# Tell all nodes when the simulation ends
#
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(stop).002 "$node_($i) reset";
}


$ns_ at  $val(stop).02 "puts \"NS EXITING...\" ; close $tracefd; $ns_
halt"

puts  "nn $val(nn) x $val(x) y $val(y) rp $val(adhocRouting)"
puts  "prop $val(prop) ant $val(ant)"
```

Fig. 24. Continued.

```
# run the simulaiton
puts "Starting Simulation..."
$ns_ run
```

Fig. 24. Continued.

The script that controls requests and agents creation is shown below.

```
# create CA servers
for {set i 0} {$i < $val(servers) } {incr i} {
      set moca_($i) [new Agent/Cacman "S"]
      $ns_ attach-agent $node_([expr $val(nn) - $i - 1]) $moca_($i)
      # let the clients print
      addTracer $moca_($i)
      # it may need it!
      #$moca_($i) cacman2dsr
      $moca_($i) cacman2aodv
}

# special case for COCA
if { $simulation_mode == "COCA" } {
   puts "COCA mode adding servers to cocas";
# in COCA case also add them each athor
   for {set i 0} {$i < $val(servers) } {incr i} {
     for {set j 0} {$j < $val(servers)} {incr j} {
      $moca_($i) AddMocas $moca_($j)
     }
   }
}



# new radnom generator requests
set r [new RNG]

# create clients
for {set i 0} {$i < [expr $val(nn) - $val(servers)] } {incr i} {
   # the extra param after C is used by cacman only
   set cac_($i) [new Agent/Cacman "C" ]
   if { $simulation_mode == "CACMAN" } {
      # specify cache size and type
      #$cac_($i) createcache [expr int($val(nn)* 0.30 * $val(thresh)) ]
      $cac_($i) createcache [expr int($val(nn)* 0.50 ) ]
      #$cac_($i) createcacheColored [expr int($val(nn)* 1.00 ) ] "8" ;
   }
   $ns_ attach-agent $node_($i) $cac_($i)
   $cac_($i) cacman2aodv
   # this is also correct $node_($i) attach $cac_($i)
   for {set j 0} {$j < $val(servers)} {incr j} {
      $cac_($i) AddMocas $moca_($j)
   }
```

Fig. 25. Scenario configuration script.

```
    # let the clients print
    addTracer $cac_($i)
}


# how many requests will be sent
if { $val(nn) >= 300 } {
    set totalreqs 200
} elseif { $val(nn) == 150 } {
    set totalreqs 100
} else {
    puts "Warning you are using not standard value\nall clients will
send 10 rquests"
    set totalreqs [expr $val(nn) - $val(servers)]
}

# schedule 1000 reqeusts (i.e. 10 CREQ/client)
for {set i 0} {$i < $totalreqs } {incr i} {
    for {set j 0} {$j < 10} {incr j} {
        $ns_ at [$r uniform 0 $val(stop)] "$cac_($i) CREQ"
   }
 }
```

Fig. 25. Continued.

B.IV CACMAN Certificate Reply (CREP) Implementation

```
void CacmanClientAgent::recv (Packet *pkt, Handler *) {
  // Access the IP header for the received packet:
  hdr_ip *hdrip = hdr_ip::access (pkt);
  // Access the Cacman header for the received packet:
  hdr_cacman *hdr = hdr_cacman::access (pkt);
  // cacman should not inspect a packet the does not belong to it
  int s = hdrip->src_.addr_ >> Address::instance().NodeShift_[1];
  int d = hdrip->dst_.addr_ >> Address::instance().NodeShift_[1];
  double now =Scheduler::instance().clock();
  double rtt = (now - hdr->send_time) * 1000;
  //jitter delay
  const double response_delay = .640;
  //RNG rdelay;

  if (s == this->here_.addr_) {
      // Discard the packet its a loop
        cerr << sim_mode_Str << "("<< this->here_.addr_<<")"<<"a
loop\n";
  } else if (d == this->here_.addr_ && hdr->ret == 1){
// This packet is addressed to me and it has a respose for my request
// so add it to the cache manager
        shareCache->addShare(hdr->cert_id , hdr->share_id);
        if(shareCache->find(hdr->cert_id,COMPLETE_CERT) || hdr-
>share_id == COMPLETE_CERT) {
```

Fig. 26. CACMAN X-Rep CREP implementation.

```
                //fill it as we got 't' shares
                for (int i =0 ; i < threshold ; i++)
                      certCache_[hdr->send_time].insert(i);
          };
          if(hdr->share_id != COMPLETE_CERT){
          //this to show how many replies has been acctually collected
                certCache_[hdr->send_time].insert(hdr->share_id);
          };
          //prepare output to trace file
          ost << sim_mode_Str <<" @" <<now<< " Node("<<this-
>here_.addr_<<") from ("<<s<<") to ("<<d<<") CREP for request#("<<hdr-
>send_time<<") cert("<<hdr->cert_id <<") rtt("<<rtt<<")
("<<certCache_[hdr->send_time].size()<<" out of "<<threshold<<")" ;
          ost.put(0);
        trace ("%s",ost.str().c_str());
          ost.str("");
    } else if (d == this->here_.addr_ && hdr->ret == 0){
           //I've been asked directly
           set<int> pool = shareCache->findAll(hdr->cert_id);
           //send all my records about this cert
           if (!pool.empty()) {
                  //send only one
                  int foo = 0;
                  for (set<int>::iterator it = pool.begin() ; it !=
pool.end() && foo < 1; ++it, foo++) {
                  if(verbose && 1) cerr << sim_mode_Str<<"
Node("<<this->here_.addr_<<") cache hit for c("<<hdr-
>cert_id<<","<<*it<<")\n" ;
                        ost<<sim_mode_Str<<" Node("<<this-
>here_.addr_<<") from ("<<s<<") to ("<< d << ") CREP for request#(" <<
hdr->send_time << ") send cert(" <<hdr->cert_id<<","<<*it<<")";
                        ost.put(0);
                        trace ("%s",ost.str().c_str());
                        ost.str("");
                        // Create a new packet
                        Packet *spkt = allocpkt ();
                        // Access the Cacman header for the new packet:
                        hdr_cacman *shdr = hdr_cacman::access (spkt);
                        //set response
                        shdr->ret = 1;
                  // Store the current time in the 'send_time' field
                        shdr->send_time = hdr->send_time;
                        // put the cert rep we have
                        shdr->cert_id = hdr->cert_id;//cert;
                        shdr->share_id = *it;
                        hdr_ip *hdrretrip = hdr_ip::access (spkt);
                        hdrretrip->dst_.addr_ = s;
                        hdrretrip->src_.addr_ = this->here_.addr_;
                        hdrretrip->dport() = hdrretrip->sport();
                        // Send the packet
                        send (spkt, 0);
                        // experimental comment send above if used
                        //Scheduler::instance().schedule(target_, spkt,
response_delay * rdelay.uniform(1.0));
```

Fig. 26. Continued.

```
                }
            }
  } else if (d >= ALL_SERVERS || d == IP_BROADCAST){
        //this is a flood or all servers request check if I can help
        if (reqs.find(hdr->send_time) == reqs.end() && rng.uniform(1.0)
> 0.0) {
    //handle duplicate requests, this request has not seend before
    //remember it
        reqs.insert(hdr->send_time);
          set<int> pool = shareCache->findAll(hdr->cert_id);
            int share = -2; //any neg number except -1 is fine
            //How many reps (51) means X-Rep-All
            for (int i=0;i<51;i++) {
                  if (!pool.empty()) {
                        if (pool.find(-1) != pool.end()) {
                              share = COMPLETE_CERT;
                        } else {
                        int random_pos = rng.uniform((int)pool.size());
                        //bad way to do it! but this seems the only way
                              set<int>::iterator it = pool.begin();
                              for (int i=0 ;i < random_pos; i++)
                                    it++;
                              share = *it;
                              pool.erase(share);
                        }
                        ost<<sim_mode_Str<<" Node("<<this-
>here_.addr_<<") from ("<<s<<") to ("<< d << ") CREP for request#(" <<
hdr->send_time << ") send cert(" <<hdr->cert_id<<","<<share<<")";
                        ost.put(0);
                        trace ("%s",ost.str().c_str());
                        ost.str("");
                        // Create a new packet
                        Packet *spkt = allocpkt ();
                        // Access the Cacman header for the new packet:
                        hdr_cacman *shdr = hdr_cacman::access (spkt);
                        //set response
                        shdr->ret = 1;
                  // Store the current time in the 'send_time' field
                        shdr->send_time = hdr->send_time;
                        // put the cert we have
                        shdr->cert_id = hdr->cert_id;//cert;
                        shdr->share_id = share;
                        hdr_ip *hdrretrip = hdr_ip::access (spkt);
                        hdrretrip->dst_.addr_ = s;
                        hdrretrip->src_.addr_ = this->here_.addr_;
                        hdrretrip->dport() = hdrretrip->sport();
                        // Send the packet
                        send (spkt, 0);
                        //experemental
                        //Scheduler::instance().schedule(target_, spkt,
response_delay * rdelay.uniform(1.0));
                  }
```

Fig. 26. Continued.

```
                    //we need to send the complete cert once!
                  if (share == COMPLETE_CERT) break;
              }
          }
  } else {
        // to be completed
       //if(verbose)
       cerr << sim_mode_Str << this->here_.addr_ <<" This packet is not
addressed to me but I'm having it s="<<s<<" d="<<d<<"\n" ;

  };
  //at last! done from the packet
  Packet::free (pkt);
}
```

Fig. 26. Continued.

APPENDIX C

ACRONYMS

TABLE X
IMPORTANT ACRONYMS USED IN THE THESIS AND THEIR MEANINGS

Acronym	*Meaning*
AO	Adjusted Overhead
AODV	Ad-hoc On-Demand Distance Vector Routing
CA	Certificate Authority
COCA	Cornell Certificate Authority
CREP	Certificate Reply
CREQ	Certificate Request
CRL	Certificate Revocation List
DSR	Dynamic Source Routing
FOR	Flooding overhead per request
MANET	Mobile Ad hoc Network
MOCA	Mobile Certificate Authority
NS	Network Simulator
PGP	Pretty Good Privacy
PKI	Public-Key Infrastructure
POSR	Packet overhead per successful request
PSS	Proactive Secret Sharing
URSA	Ubiquitous and Robust Access Control for Mobile Ad Hoc Networks
RSA	Rivest, Shamir, and Adelman

VITA

Laith A. Al-Sulaiman was born in Norman, OK, on September 28, 1976. He received his Bachelor of Science in Computer Science from the Faculty of College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia, in June 1999. He worked as a Teaching Assistant for the Department of Computer Science at Imam Muhammad bin Saud Islamic University from 1999 to December 2000. In December 2002, he received his Master of Science from the Department of Computer Science, Faculty of Science, Old Dominion University. He started working on his Ph.D. degree in Computer Science at Old Dominion University, in January 2003. During the course of his Ph.D. study, he co-authored three scientific papers and technical reports. He is a member of IEEE, IEEE Computer Society, and ACM.