A Privacy-Preserving Cache Management System for MANETs

Kassem Fawaz, Noor Abbani, Hassan Artail
Department of Electrical and Computer Engineering
American University of Beirut
Beirut, Lebanon
{kmf04, nma51, hartail}@aub.edu.lb

Abstract— **Mobile Ad hoc Networks (MANETs) have become increasingly popular with the rapid emergence of hand-held devices and advanced communication technologies. As a result, several MANET applications have been proposed one of which is the data access application. To enhance the performance of this application cache management systems have been suggested; however, they have been designed regardless of the privacy concerns they raise. We study the cache management system COACS (a COoperative and Adaptive Caching System for MANETs) and its weaknesses in terms of privacy to propose a privacy-preserving protocol to render such a caching system well protected against all kind of internal or external privacy breaches. We also provide a mathematical analysis to measure the system's degree of anonymity.**

Keywords: Privacy, anonymity, cache management systems, MANETs

I. INTRODUCTION

With the increasing prevalence of mobile handheld electronic devices and the rapid advances in communication technologies, Mobile Ad Hoc Networks (MANETs) have become increasingly popular. Contrary to infrastructure based networks, MANETs have no fixed routers or base stations; therefore, most applications require the cooperation of nodes in forwarding and routing each other's packets which exposes the data to other nodes leading to a privacy breach. In addition to data exposure due to node collaboration, MANETs are also subjected to possible network access and hence passive traffic analysis or active packet manipulation.

One well-known ad hoc application is data access where nodes collaborate to access a database server by sending data item queries. The need to reach the database server for each query renders this application costly in terms of delay, power consumption and bandwidth utilization. Therefore, because MANETs are resource-limited networks, caching data has been given much attention with the goal of limiting the depletion of the mentioned resources.

Several systems have been proposed for cache management in MANETS, none of which has addressed privacy as a critical issue making it majorly compromised in such frameworks as will be detailed later.

Initially, CacheData and CachePath were introduced as methods where nodes store the data for a query that is frequently requested or caches the address to a caching node respectively. They allow the caching node and the database server to know the source of the request, and hence deduce user interests. Other approaches in [3], [4] and [5] divide the network into zones or clusters; when a node receives a request, it broadcasts the request if not found in its local cache. This also links the requesting node with the data which allows nodes or the data server to profile users.

Similarly in [1], a cache management system, named COACS (cooperative and adaptive cache system), is proposed where nodes play the role of a cache node (CN) or a query directory (QD). If a requesting node (RN) receives the response from the server i.e., the item is not already cached in one of the CNs, it caches the data locally. Therefore, there is a clear association that a node caching data has requested that data at some point in time. This leads to the breach in privacy of the node's request and interests.

As can be deduced, privacy concerns have not been addressed in caching systems for MANETs. Hence, in this paper we present a protocol that implements privacy mechanisms for the COACS framework using techniques that include source anonymity, encryption and request hopping. With these approaches we aim at having a privacy preserving collaborative and adaptive cache management system.

The paper is organized as follows: Section II gives an overview of COACS, while Section III provides work related to cache managements systems and privacy in MANETs. In Section IV we describe the system design, which we analyze in section V. Finally, future work and the conclusion are presented in section VI.

II. BACKGROUND

As described in [1], in COACS, the system's operations commence when a node wants to request an item prior to the formation of the caching system. The initial requesting node initializes the system by electing the QDs, where QD election takes place according to a score criterion that summarizes a node's resources and capabilities.

After the QD election, when an RN requires data, it sends a data request packet (DRP) to the nearest QD. If the receiving QD finds the data requested (QD 2 in Figure 1), it will send the request to the involved CN (CN 6), which will directly send the data to the RN in a data reply packet (DREP). Otherwise, if the QD does not find the data in its index (QD1), it will append its address to the packet and forward the request to the nearest QD (QD 2 in Figure 1), which will go through the same process. Finally, if all QDs are checked, the data is finally forwarded to the data source which will send the data to the RN that will have to cache the data since it has not been cached on any other node.

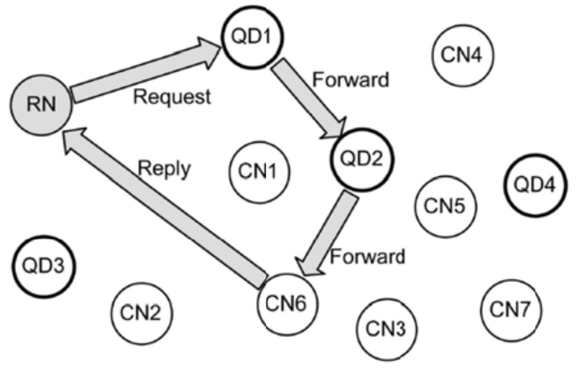

Figure 1: COACS Scenario of an RN requesting data cached in CN6 [1]

III. RELATED WORK

A. *Cache Management Systems*

All cache management systems for MANETs proposed so far have weaknesses in terms of privacy. In [2], if the data item is not in the requesting node's local cache, it checks a table of recent requests that records which node has requested which items, and the request is forwarded to the nearest node. This information is collected using the zone broadcasts that other nodes issue when they fail to find the item in their recent request table. It follows that this protocol provides no countermeasures to privacy invasion. Both the data source and nodes on the path to it can collect user request information. More importantly, all nodes, by the design of the zone cooperation mechanism, can collect information about all the nodes in their zone.

Other protocols were based on dividing the network into zones in [4], or clusters in [3], [5] and [6]. Again, the node first checks if the data is in its local check, and if it is not, it broadcasts the request to the zone members or the cluster head. Therefore effectively upon each request to a new item in the network, a very high number of nodes including the cluster head will know about the cache request and consequently can associate users to data item requests.

As can be understood from the brief survey, cache management systems have been proposed oblivious of the privacy flaws and breaches they incur. However, there is considerable research on the topic in MANETs, especially in the area of routing. In the next subsection, the approaches for preserving privacy in MANETs are speculated.

B. *Privacy in MANETs*

In [9], the authors implement variable security levels and suggest asymmetric digital signatures to provide data confidentiality and authentication, so that the users are able to hide their sensitive information from other users.

On the other hand, the authors in [10] target sender anonymity in web browsing by introducing a novel method called *Crowds*. Each node has a set of *N* nodes which constitute its ambiguity set, and randomly decides to send the message to the real destination or forwards it to another node within the set. This way, the destination node cannot tell whether the source is the actual sender or a random node, which applies to any node on the "path". The destination node replies back to the sender in the reverse path.

In [11], the authors propose a source anonymous authentication scheme that aims to conceal the source identity but authenticates him at the same time. The sender constructs its message, concealing it in *N* nested messages, where each member of the ambiguity set unwraps one layer, until the message reaches its final destination that is included within the message implicitly. The reply process is not clear, but is assumed to follow the same path as the requestor as in Crowds. This approach requires a high traffic overhead, which might be the cost of providing privacy. In our case however, sender authentication is not needed since, as we argue later, sender or source privacy is what we need along with some smart caching protocol design.

From a totally different perspective than that of ambiguity sets, in [12], the focus is on the networking layer (routing) where it is proposed to use pseudonyms and random identifiers that change with time. These identifiers are used by nodes to replace the source and destination addresses in the control and data traffic. This approach makes it difficult to sustain long living routes and makes it infeasible to have proactive routing protocols. Such a protocol also renders DSDV unusable, and therefore, another reactive routing protocol needs to be used due to changing identifiers.

In summary, and contrary to the above approaches, our privacy-preserving mechanisms implemented on COACS do not affect its usability with all routing protocols while maintaining its performance acceptable.

IV. SYSTEM DESIGN

It is of great importance, when aiming for privacy or security in a particular system, to know the details of the system and its architecture. Moreover, it is vital to study the threats on this system and the possible privacy breaches that its nodes and data will be exposed to.

A. *System Model*

The system consists of a MANET of wireless mobile nodes that are interested in certain data generated at a data source. The data source (server) abstracts different sources, and is connected to the MANET via a gateway through a wired network as depicted in Figure 2.

We assume that all nodes cooperate to provide both the caching and privacy systems for other participating nodes. Nodes participate in caching and privacy mechanisms and provide these services to other nodes on the account that these other nodes provide them with similar services. Moreover, we assume that there exists a routing protocol that enables efficient multi-hop communications between nodes in the MANET. We finally assume that a request message is fixed and small in size relative to a response message that is usually composed of packet streams.

In addition to the system assumptions, we assume that there exists a trustworthy public key management system that maintains and distributes the public key for each node. As a result, each node has the public keys for all other nodes before the system operates. Moreover, this repository maintains the public keys for the newly introduced nodes in the system. Furthermore, we assume that the QDs share a group key, by which a request can be encrypted. A shared group key between the QD nodes is needed since the same request might traverse several QDs, and these QDs must be able to decrypt the encrypted request. This scheme requires less overhead than encrypting the requests with the key of each QD. Finally, we assume that there exists a mechanism to maintain the group key of the QDs in case of the insertion or deletion of a QD.

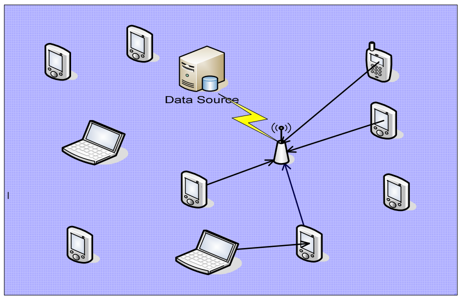

Figure 2: General System Model of a Data Access MANET Application

B. *Adeversary Model*

Different attacker properties are defined in [13], where the authors organize them in three orthogonal dimensions. The first dimension is whether the attacker is an internal node in the system or an external node. An internal node in the system is one that cooperates within the framework of COACS, such as a CN or a QD. The second dimension covers the attacker activity, whether passive or active. A passive attacker relies on gathering and analyzing gathered information without interfering in the system's operations by sending or modifying packets, as the case with active attackers. Finally, in the third dimension, an attacker can be local by monitoring limited aspects of the network, or global by having access on the whole network communications.

In this system, we assume the adversary to be a hybrid combination of local internal passive and global external passive attackers. The first attacker encompasses some compromised internal nodes such as the CNs and QDs that passively monitor passing messages. While the second attacker can monitor traffic flows and is able to infer which nodes are communicating with each other. We further assume that all nodes in the system attempt to gather as much information as they can from passing traffic, although they are not malicious and don't collude with other nodes.

We also assume that these two attackers collude with each other. As a result, the external adversary can eavesdrop to track messages based on their sequence numbers and collude with certain nodes in the system to tell their actual content. It is worth noting that passive attackers are harder to detect, more critical, and more relevant to privacy infringements than the active attackers. However, we assume that this attacker has limited computational capability and thus cannot break through cryptographic measures.

C. *Source Anonymity*

In order to prevent the profiling of users according to the requested data items, it is essential to hide the source of the request. The source anonymity scheme protects the identity of the RN in both the request and reply paths. The ***request path*** is defined as the set of nodes that forward the request from the RN to the first QD, while the ***reply path*** is the set of nodes that forward the request/response from the QD back to the RN, including the intermediary QDs and CN. The system prevents the attacker, the recipient QDs and CN from telling the source of the query or the destination of the response in request and reply paths respectively.

In the request path, we utilize an approach similar to Crowds in [10]. In our context, the cooperating nodes in the system which all the nodes are aware of constitute the set of "jondos". However this approach does not protect users from local eavesdroppers or from global attackers monitoring communication flows. For this purpose we utilize two more mechanisms to enhance privacy which include piggybacking the request on passing requests or responses, in addition to encrypting the request and changing its form each hop.

D. *Request Piggybacking*

The global attacker can easily expose the source as the node that sent a COACS message after being idle for some time and in response to no specific event. Therefore, it is imperative to conceal the actual source of the request, through hiding the request event itself. The adversary can also monitor the response message stream passing through the source node, and thus, using timing analysis it can narrow down the response message targeted for this source. Moreover, encrypting the response message will not benefit either, as it is virtually impossible to hide the destination from the global attacker because is related to the source anonymity problem.

Most of the existing techniques resort to dummy request generation to hide the sender among a set of the senders, and to hide the actual request among fake requests. This scheme results in a considerable amount of overhead traffic and battery consumption that cannot be accommodated in a MANET environment.

In our system, the message will travel several hops before reaching the intended destination. Hops stand for the Crowds "jondos" that are multi-hop away in routing layer terms. As a result, request and response traffic will cover many nodes on the way to the destination, and other nodes can benefit from this traffic to piggyback their requests. A piggybacking scenario is highly probable, as nodes forward traffic for other nodes, act as QDs and CNs so they will be routing requests and responses most of the time, and an RN can afford to wait a little amount of time till a request or response message is passing through. Specifically, each node has an average of the interval between consecutive requests to be forwarded through. We refer to this average as the *inter-request interval* T_{avgreq}. Also it maintains an average of the interval between the time this node routed a request and the time it routed the corresponding response referred to as the *inter-response interval* T_{avgrsp}.
Those statistics allow the node to predict when a response or request will pass through it, giving it the possibility of piggybacking its request in case it had one. An RN wanting to request a data item will check the inter-request and inter-update intervals against the last request and check if within a period of T_{th} *ms* it predicts that a request or a response will pass through. This condition is shown below and referred to as *Event A* in the rest of the paper:

$$T_{avgrsp} - (\mathrm{t_{reqneed}} - \mathrm{t_{lstpcktrcvd}}) < T_{th}$$

If so, the RN waits and then piggybacks the request on whatever message comes through. It is worth noting, that a node piggybacks its own requests only and never combines requests coming from different directions.

Moreover, all request messages are of constant sizes and can accommodate up to ***K*** requests for data items in the same message. If a request is piggybacked as it is, the global attacker will be able to detect a change in the size of the packet and deduce that a request was piggybacked. Therefore, the packet size will be kept constant as will be explained in the hop modification section to ensure that piggybacking small sized requests on request messages and response streams will pass undetected by the global adversary monitoring the communication flows in the network, and the RN will not be detected as the source of a request.

E. Request Hopping

In this system, each request must hop over at most N nodes before reaching the QD to handle the request. Setting an upper limit helps saving redundant hops which reflect in overhead traffic and query delay. The RN piggybacks the request on a passing request or response message and it has to set the next destination of the piggybacked message which depends on the forwarding probability P_f.

The behavior of an RN is the same as any node on the request path. Of course, the delay constraints on a forwarding node are more stringent than a requesting node to avoid having unnecessary delays in the forwarding process. So, each node on the forwarding path, including the RN, after piggybacking the request if available, tosses a biased coin. This node with a probability P_f chooses to forward the request to the nearest QD, and with a probability $1- P_f$ chooses another node to forward the request to depending on the message. The probability P_f is a system parameter, where a lower probability means that the request will traverse more nodes to get to the QD which indicates a higher privacy requirement with the cost of higher delay on other hand.

This way, neither a node on the request path nor the recipient QD can tell whether the request came from the RN or from another node on the path. Each request message contains a count value that indicates how many hops it traveled. This count value is part of the request message and is encrypted so that the adversary will not have access to it. The originating RN fills this count value with a random number between 0 and *M*. Each node on the path increments the count value if it decides to send to another destination other than the QD. When the count value reaches *N*, the corresponding forwarding node sends the request directly to the nearest QD as depicted in Figure 3.

As for the case when the forwarding node chooses not to send the request to the QD, it has to find another node to send the request to. If *Event A* is satisfied the node will wait for a predicted response stream and piggybacks the request. If not, it chooses a random node and forwards the request.

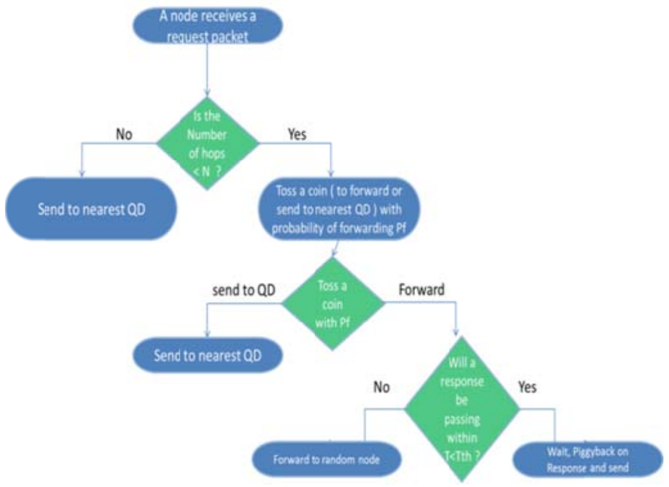
A node receives a request packet
Is the Number of hops < N ?
No
Yes
Send to nearest QD
Toss a coin (to forward or send to nearest QD) with probability of forwarding Pf
send to QD
Toss a coin with Pf
Forward
Send to nearest QD
Will a response be passing within T<Tth ?
No
Yes
Forward to random node
Wait, Piggyback on Response and send

Figure 3: The process upon the reception of a request packet

Each node on the request path saves the sequence number of the request message along with the source address of the node that forwarded the message and the previous hop (the node that the sent to the sender node) so that each forwarding node maintains two-hop information that will help in case there is node disconnection in the reply path. This will enable the nodes to route the response in the same path the request passed through, without the intermediary nodes, CN, QDs, or the global attacker knowing the intended destination of the response. Each node on the request path stores inside the response record a timeout value that is inversely proportional to the count value, and acts as an additional reliability measure. The timeout value is inversely proportional to the count value to ensure that the nodes nearer to the QD in the reply path timeout before those that are farther. The node operations are summarized in Table 1.

Node Operations

```
Initialize $t_{lastTimeRqstRcvd}$ =0
Constant $T_{avgRqstTime}$

DataStructure ReplyPathInformation
while( true)
{
    if ( a request packet is received)
      {- update $t_{lastTimeRqstRcvd}$
       - Append Address
       - store sequence number
         with addresses of nodes of two
         previous hops in ReplyPathInformation
       - Check if Request is waiting to be
         Piggybacked
       - Piggyback request if available
       - Forward packet to another node with $P_f$ or to
         QD with 1-$P_f$}

    if ( a reply packet is received)
      {- Check Sequence Number
        - Forward packet to next node from
          ReplyPathInformation for this sequence
          number
        - If node disconnected send to another node
          stored in ReplyPathInformation for this
          sequence number
       }
}
```

Table 1: Node Operations

F. Request Encryption and Hop Modification

One major premise in the system is to be able to piggyback node requests on top of request messages. This requires the request message to change form at each hop and keep the same size as well. Each time a message hops it will change form and thus an attacker will not be able to detect if new requests were piggybacked. Each request message is composed of several requests that are less than K and random padding to keep the size constant. So, when a node piggybacks a request it removes part of the padding and adds its own request and sends the message again. This will prevent an attacker from detecting if an item was piggybacked or it is a normal changing form of the packet. On the other hand, a local eavesdropper might snoop into the message content and view the requested data items. For this purpose, the request must traverse the reply path encrypted, and it must be re-encrypted at each node at the forwarding path to ensure changing form.

To start with, each RN encrypts its own request with the shared public key of the QDs, and can either send it to another node or piggybacks it on a request message or response stream. The RN then generates a random key and performs symmetric encryption on the request message. Then it encrypts this symmetric key along with other header information including a random nonce to prevent replay attacks with the public key of the next hop. Then, when the next hop receives the message, it decrypts the header with its private key and gets the count value and the symmetric key. It then can add its own request if available, or just change the random padding. This node then repeats the above process. This way, the message headers will be protected and the whole message will change form at each hop, preventing the adversary from being able to tell if a forwarding node is a requesting node as well.

The node performs asymmetric key encryption on the request itself which is small in size; albeit an inefficient process, it does not make difference on small sized data. However, we resort to symmetric key encryption on the request message that encompasses many requests. Finally, to prevent the overhead of setting symmetric keys between all nodes in the system, the node sends the key in the request encrypted with the recipient's public key.

V. ANONYMITY ANALYSIS

To study the degree of anonymity in our system, we will perform a preliminary mathematical analysis to have an insight on the performance of the system in terms of privacy. In [14], the authors provide metrics for Crowds with a system of N nodes, and C nodes collaborating with the attacker.

The probability that the sender is the predecessor of the first collaborator on the path is given by:

$$p_1 = 1 - P_f\left(\frac{N-c-1}{N}\right) \qquad (1)$$

Consequently, the probability that the users are the senders of the message is given in Equation 2:

$$p_i = 1 - \frac{1-p_1}{N-c-1} = \frac{p_f}{N} \qquad (2)$$

However, in our system, the probability given to the predecessor of the collaborating node depends also on the probability that it has piggybacked a request:

$$p_1 = 1 - P_f\left(\frac{N-c-1}{N}\right) - p(pigbkng) \qquad (3)$$

The probability of piggybacking depends on the probability of the node having a request and on the probability that event A is satisfied as mentioned in the piggybacking section. Assuming that these two events are independent, the probability of piggybacking is:

$$p(pigbkng) = p(req)\,p(EventA) \qquad (4)$$

Since requests passing through a node are events that occur continuously and independently at a constant average rate, the event of request arrival can be modeled as a Poisson process. Consequently, the inter-arrival time between requests follows an exponential distribution: $\frac{1}{T_{avgreq}} e^{\frac{-t}{T_{avgreq}}}$

We will study the effect of $T_{avgrequesttime}$, T_{th} on the above probability by studying its effect on $P(event\ A)$.

$$\begin{aligned} p(EventA) &= p\left(T_{avgreq} - \left(t_{reqneed} - t_{lstpcktrcvd}\right) < T_{th}\right) \\ &= p\left(T_{avgreq} < T_{th} + t_{reqneed} - t_{lstpcktrcvd}\right) \qquad (5) \\ &= p\left(T_{avgreq} < T_{th} + T_{cst}\right) = 1 - e^{-T_{avgreq} \times (T_{th} + T_{cst})} \end{aligned}$$

Equation 5 is the result of the cumulative distribution of a random variable that follows an exponential function. Since T_{cst} depends on the time the request was needed and the last time a packet was received, we set it to 0.1*s* to take the worst case scenario (being a very short interval), and so the waiting time is expected to be higher. The results are shown in Figure 5, where it is clear that as T_{avgreq} increases the probability of event ***A*** occurring decreases and hence piggybacking decreases. This is due to the fact that a smaller T_{avgreq} indicates less traffic and hence less messages passing through the node. This decreases the possibility of piggybacking.

Moreover, as T_{th} increases the probability of Event *A* occurring increases. This is because the node is allowed to wait more for a request to pass through it. However, having a high T_{th} incurs delays and degrades the performance in terms of latency. Our aim is to increase *p*(Event A), which as a result will decrease p_1. Decreasing the probability p_1 makes it closer to the other probabilities given to the other nodes and hence more confusing for the attacker to know the sender.

From the results, increasing ***P***(EventA) requires a low T_{avgreq} i.e., considerable amount of traffic. This traffic is made possible with the presence of request hopping. It also requires a moderate T_{th}; for example, if T_{avgreq} = 0.2 s, a T_{th} of 0.2 *s* gives a *P*(event A) of 0.8 which is acceptable to decrease p_1.

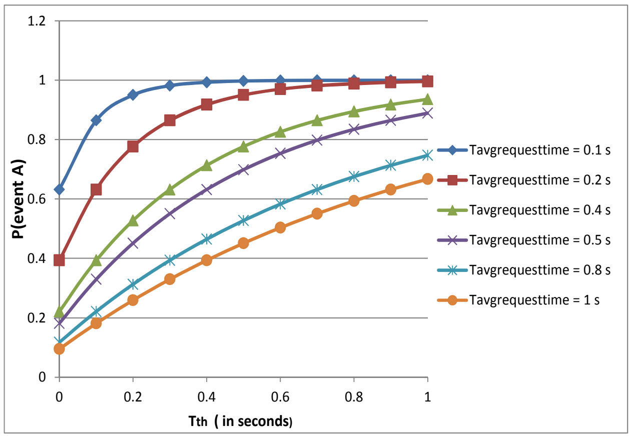

Figure 4: Probability of Event A as a function of T_{th} and T_{avgreq}

VI. CONCLUSIONS AND FUTURE WORK

All proposed cache management systems for MANETs have been designed oblivious of the privacy concerns they raise. In this paper we have designed a privacy framework for COACS to make it a privacy preserving cache management systems. It provides source anonymity against three adversaries. It protects against the global attacker by hiding actual request events through piggybacking requests on passing messages. Second, it protects the RN from the QDs, CN, or the server handling the response by having the request hop over several nodes before reaching its destination. Finally, it protects against the local eavesdroppers and global attackers by encrypting the requests and changing their forms on each hop to prevent exposing the content and tracing the request.

For future work, we plan next to implement this system using the network simulator OPNET or NS2. Such a simulation is expected to provide us with concrete results of this system in terms of reliability, performance, latency and anonymity. We also plan to further generalize these operations to fit several cache management systems other than COACS.

REFERENCES

[1] H. Artail, H. Safa, K. Mershad, Z. Abou Atme and N. Suleiman, "COACS: A Cooperative and Adaptive Caching System for Manets", *IEEE Transactions on Mobile Computing*, Vol. 7, 2008

[2] Y. Du and S. K. S. Gupta , "COOP – A cooperative caching service in MANETs", *IEEE International Conference on Networking and Services*, 2005

[3] Y. Li and L. Gruenwald, "A Caching Model for Real-Time Databases in Mobile Ad-Hoc Networks", Springer Database and Expert Systems Applications, 2005

[4] N. Chand, R. C. Joshiz and M. Misra "A zone co-operation approach for efficient caching in mobile ad hoc networks", *International Journal of Communications System,* 2006

[5] N. Chand, R. C. Joshi and M. Misra, "An Efficient Caching Strategy in Mobile Ad Hoc Networks Based on Clusters", *IFIP International Conference on Wireless and Optical Communications,* 2006

[6] G. Chiu and C. Young, "Exploiting In-Zone Broadcasts for Cache Sharing in Mobile Ad Hoc Networks", *IEEE Transactions on Mobile Computing,* 2009

[7] S.Lim, W.Lee, G. Cao and C. Das, "A novel caching scheme for improving Internet-based mobile ad hoc networks performance", *ELSEVIER Adhoc Networks,* 2006

[8] C.Chow, H. Leong and A. T. S. Chan, "Group-based Cooperative Cache Management for Mobile Clients in a Mobile Environment", *IEEE International Conference on Parallel Processing"* 2004

[9] G. Cao, L.Yin and C. R. Das, **"**Cooperative Cache-Based Data Access in Ad Hoc Networks", *IEEE Computer Society,* 2004

[10] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity For Web Transactions", *ACM Transactions on Information and System Security*, 1998

[11] J. Ren, Y. Li, and T. Li, "SPM: Source Privacy for Mobile Adhoc Networks", *EURASIP Journal on Wireless Communications and Networking,* 2010

[12] H. Choi, P. McDaniel and T. F. La Porta, "Privacy Preserving Communication in MANETs**",** *IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks,* 2007

[13] C. Diaz, S. Seys, J. Claessens, and B. Preneel, "Towards Measuring Anonmity", *Proceedings of the 2nd international conference on Privacy enhancing technology,* 2002

[14] N. Jaggi, U. MarappaReddy and R. Bagai, "A three-dimensional Approach Towards Measuring Soruce Anonymity", *IEEE First International Workshop on Security in Computers, Networking and Communications.*